Optimum Allocation of Inspection Stations in Multistage Manufacturing Processes by Using Max-Min Ant System

SHETWAN, ALI,GASSIM,M.

Optimum Allocation of Inspection Stations in Multistage Manufacturing Processes by Using Max-Min Ant System

Ali Gassim Mohamed Shetwan

A thesis submitted for the degree of

Doctor of Philosophy

School of Engineering and Computing Sciences

Durham University

UK

2013

DEDICATION

To my parents and grandparents

Optimum Allocation of Inspection Stations in Multistage Manufacturing Processes by Using Max-Min Ant System

Ali Gassim Mohamed Shetwan

Submitted for the degree of Doctor of Philosophy

Abstract

In multistage manufacturing processes it is common to locate inspection stations after some or all of the processing workstations. The purpose of the inspection is to reduce the total manufacturing cost, resulted from unidentified defective items being processed unnecessarily through subsequent manufacturing operations. This total cost is the sum of the costs of production, inspection and failures (during production and after shipment). Introducing inspection stations into a serial multistage manufacturing process, although constituting an additional cost, is expected to be a profitable course of action. Specifically, at some positions the associated inspection costs will be recovered from the benefits realised through the detection of defective items, before wasting additional cost by continuing to process them.

In this research, a novel general cost modelling for allocating a limited number of inspection stations in serial multistage manufacturing processes is formulated. In allocation of inspection station (AOIS) problem, as the number of workstations increases, the number of inspection station allocation possibilities increases exponentially. To identify the appropriate approach for the AOIS problem, different optimisation methods are investigated. The MAX-MIN Ant System (MMAS) algorithm is proposed as a novel approach to explore AOIS in serial multistage manufacturing processes. MMAS is an ant colony optimisation algorithm that was designed originally to begin an explorative search phase and, subsequently, to make a slow transition to the intensive exploitation of the best solutions found during the search, by allowing only one ant to update the pheromone trails. Two novel heuristics information for

the MMAS algorithm are created. The heuristic information for the MMAS algorithm is exploited as a novel means to guide ants to build reasonably good solutions from the very beginning of the search. To improve the performance of the MMAS algorithm, six local search methods which are well-known and suitable for the AOIS problem are used. Selecting relevant parameter values for the MMAS algorithm can have a great impact on the algorithm's performance. As a result, a method for tuning the most influential parameter values for the MMAS algorithm is developed.

The contribution of this research is, for the first time, a methodology using MMAS to solve the AOIS problem in serial multistage manufacturing processes has been developed. The methodology takes into account the constraints on inspection resources, in terms of a limited number of inspection stations. As a result, the total manufacturing cost of a product can be reduced, while maintaining the quality of the product. Four numerical experiments are conducted to assess the MMAS algorithm for the AOIS problem. The performance of the MMAS algorithm is compared with a number of other methods this includes the complete enumeration method (CEM), rule of thumb, a pure random search algorithm, particle swarm optimisation, simulated annealing and genetic algorithm. The experimental results show that the effectiveness of the MMAS algorithm lies in its considerably shorter execution time and robustness. Further, in certain conditions results obtained by the MMAS algorithm are identical to the CEM. In addition, the results show that applying local search to the MMAS algorithm has significantly improved the performance of the algorithm. Also the results demonstrate that it is essential to use heuristic information with the MMAS algorithm for the AOIS problem, in order to obtain a high quality solution. It was found that the main parameters of MMAS include the pheromone trail intensity (α), heuristic information (β) and evaporation of pheromone (ρ) are less sensitive within the specified range as the number of workstations is significantly increased.

Declaration

The work in this thesis is based on research carried out within the Manufacturing, Design and Management Group, School of Engineering and Computing Sciences, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgements

I wish to thank my previous thesis adviser Prof. Valentin Vitanov for his support and advice. With great sadness that Prof. Vitanov has passed away May 2011. I will remember him as a gentle, kind man.

I very warmly thank my thesis adviser Dr. Hui Long for her support and advice. Many thanks to Dr. Peter Matthews for his support and advice who has been of invaluable help in writing this thesis. This thesis would not exist without their help. Finally, thanks to my family for their endless support.

Contents

Contents

Contents

Contents

Contents

Contents

List of Figures

List of Figures

List of Tables

List of Tables

List of Abbreviations

ACO	Ant colony optimisation
ACS	Ant colony system
AOIS	Allocation of inspection station
AS	Ant system
B&B	Branch and bound
CEM	Complete enumeration method
CMA-ES	Covariance matrix adaptation evolution strategy
DE	Differential evaluation
DFOS	Deviation from optimal solution
DP	Dynamic programming
EDAs	Estimation of distribution algorithms
ES	Evolution strategy
FDC	Fitness distance correlation
FMS	Flexible manufacturing system
GA	Genetic algorithm
GCM	General cost model
GRASP	Greedy randomized adaptive search procedure
ILP	Integer linear programming
IP	Integer programming
LP	Linear programming
MDP	Markov decision process
MMAS	Max-min ant system
MSPS	Multistage production system
NLP	Non-linear programming
PAF	Prevention-appraisal-failure
PRS	Pure random search
PSO	Particle swarm optimisation
QAP	Quadratic assignment problem
RAP	Redundancy allocation problem
ROT	Rule of thumb
SA	Simulated annealing
TS	Tabu search
TSP	Travelling salesman problem
WIP	Work in-progress
WS	Workstation

Notation for the general cost model

k	Workstation k in the manufacturing system (k=1, 2,…., n).
M	Feasible inspection stations allocation.
m	Inspection station assigned in the manufacturing system.
n	Number of processing workstations in the system.
B	Number of parts entering the system.
s_k	Sample size for workstation k.
a_k	Acceptance number for workstation k.
b_k	Number of bad items in sample of workstation k.
α_m	Probability that inspection station m incorrectly classifies CU as NCU.
β_m	Probability that inspection station m incorrectly classifies a NCU as a CU.
δ_k	Probability of repairing a defective unit at the workstation k.
P	Final sale price of each unit.
Z_k	Probability of a non-conforming of part processing at the workstation k.
Y	Direct cost of material to repair a defect at the customer's end.
g_k	Unit reworking cost at the workstation k.
u_k	Unit scraping cost at the workstation k.
W	Percentage of parts replaced at the customer's end, $W \in (0, 1]$.
IC_m	Unit inspection cost at inspection station m.
FC_m	Fixed inspection cost at inspection station m.
U_k	Unit manufacturing cost at the workstation k.
NG_k	Number of conforming parts leaving the workstation k.
ND_k	Number of defective parts leaving the workstation k.
R_k	Number of parts for reworking at workstation k.
CP_k	Number of conformed parts.
NCP_k	Number of non-conformed parts.
TIC_m	Total inspection cost at inspection station m.
SC_k	Scrapping cost at the workstation k.
RC_k	Rework cost at the workstation k.
TMC_k	Total manufacturing cost for parts at the workstation k.
IFC_k	Internal failure cost at workstations k.
EFC_k	External failure cost at workstation k.
NG^n	Number of conforming parts leaving the workstation n.
$S_{U,k}$	Score number given for the unit manufacturing cost at workstation k.
$S_{Z,k}$	Score number given for the unit defective rate at workstation k.
$S_{a,k}$	Total score number given at workstation k.

Notation for the ACO and PSO

P_{ij}	Transition probability of combination.
α	Relative importance of the pheromone trail intensity.
β	Relative importance of the problem-specific heuristic information.
ρ	Controls the evaporation of pheromone in the environment $0 < \rho \leq 1$.
q	Random number uniformly generated $q \in [0,1]$.
q_0	A parameter which determines the relative importance of exploitation versus exploration.
τ_{ij}	Pheromone trail intensity of combination (i, j).
η_{ij}	Heuristic information of combination (i, j).
Q	The total amount of pheromone released on all paths by the ants in one cycle.
p_{best}	Probability that an ant will construct the best solution.
avg	Average number of different choices available to an ant at each step while constructing a solution
$f(s_{opt})$	Optimal solution value for the problem.
τ_0	Specifies the initial pheromone level on all edges.
$\tau_{\min}$	Represents the lower limit for the pheromone trail strength.
$\tau_{\max}$	Represents the upper limit for the pheromone trail strength.
τ_{ij}^{gb}	Global best ant is allowed to add pheromone after each iteration.
λ	Sensitivity of the λ -branching factor.
J	Random variable that selects a good node from neighbourhood i.
τ_{ij}^{ib}	Local best ant is allowed to add pheromone after each iteration.
w	Number of elitist ants.
v_i^k	Velocity of particle i at iteration k.
v_i^{k+1}	Velocity of particle i at iteration $k + 1$.
ω	Inertia weight.
c_j	Acceleration coefficients; $j = 1, 2$.
$rand_i$	Random number distributed uniformly between 0 and 1; $i = 1, 2$.
x_i^k	Current position of particle i at iteration k.
$lbest_i$	Best position of particle i.
$gbest$	Position of best particle in a population.
x_i^{k+1}	Position of the particle i at iteration $k + 1$.
PS	Number of particles.
D	Distances to the optimal solution, $D = \{d_1, d_2, ..., d_n\}$.
F	Individual fitnesses, $F = \{f_1, f_2, ..., f_n\}$.
σ_F	Standard deviation of F.
σ_D	Standard deviation of D.
$\bar{f}$	Means of F.
$\bar{d}$	Means of D.

Chapter 1
Introduction

The manufacturing cost of a product is one of the major factors under consideration for manufacturing companies. Increasing product cost leads to negative effect on the overall competitiveness for these companies. The usual requirement is that products are manufactured to an acceptable quality level and at minimum cost. The total cost of a product is the sum of the costs of production, inspection, internal failures and external failures. In multistage manufacturing processes, inspection stations should then be located after some or all of the processing workstations, to guarantee that a specific quality level is being maintained. The purpose of the inspection stations is to screen out the defective items before adding extra costs by continuing to process them. Consequently, the total cost of the product can then be minimised.

This research focused specifically on avoidable costs resulting from unidentified defective items being processed unnecessarily during manufacturing operations. Also, studying the strategies employed to allocate limited inspection stations into manufacturing processes to reduce the total manufacturing cost. These strategies usually propose numerical algorithms for the allocation of an economically appropriate number of inspection stations. This can be done by finding a balance among different cost components, related to inspection, scrap, repair and replacement as a result of quality failure, and/or the warranty penalty in the case where a non-conforming product has been shipped to customers.

The objective of this research is to propose a methodology, using MAX-MIN Ant System (MMAS) algorithm to allocate number of inspection stations in a serial multistage manufacturing process. As a result, the total manufacturing cost of a product can be reduced

without affecting the quality of the product. It should be noted that the methodology may not be used as a monitoring tool to detect changes in production performance.

1.1 Background

The procedure of making decisions about whether or not to inspect a final or semi-finished product at every processing workstation in a serial multistage manufacturing process, consisting of *n* processing workstations is shown schematically in Figure 1.1. The product may transfer to the next stage or to the final consumer if there is no need to perform inspection, otherwise a product will be inspected, and the inspected products may conform or not to the predefined quality requirement. In the case of conforming items, they will be sent to the next stage. There are several possibilities in the case of non-conforming items: (i) they may be reworked and sent to the next stage; (ii) they may become a downgraded product; or (iii) they may be scrapped.

Placing inspection points in a multistage manufacturing process, although constituting an additional cost, at some level of inspection points is expected to be a profitable course of action. The associated costs will be recovered from the benefits realised through the detection of defective items. In other words, it is assumed that if inspection is performed after every processing workstation, then the scrap, replacement, downgrading, and reworking costs (internal failure costs) will be minimised, as defective items will be identified before adding extra costs to already defective material. At the same time, non-conforming items can be screened out before reaching the customer, which may result in additional costs (external failure costs). On the other hand, these savings have to be considered against the inspection costs, which include equipment, staff, time, shop floor space and create new queues in the system that might add extra work in-progress (WIP) and flow. As a result, if these in-process inspections are performed unnecessarily or too often greater costs will incur.

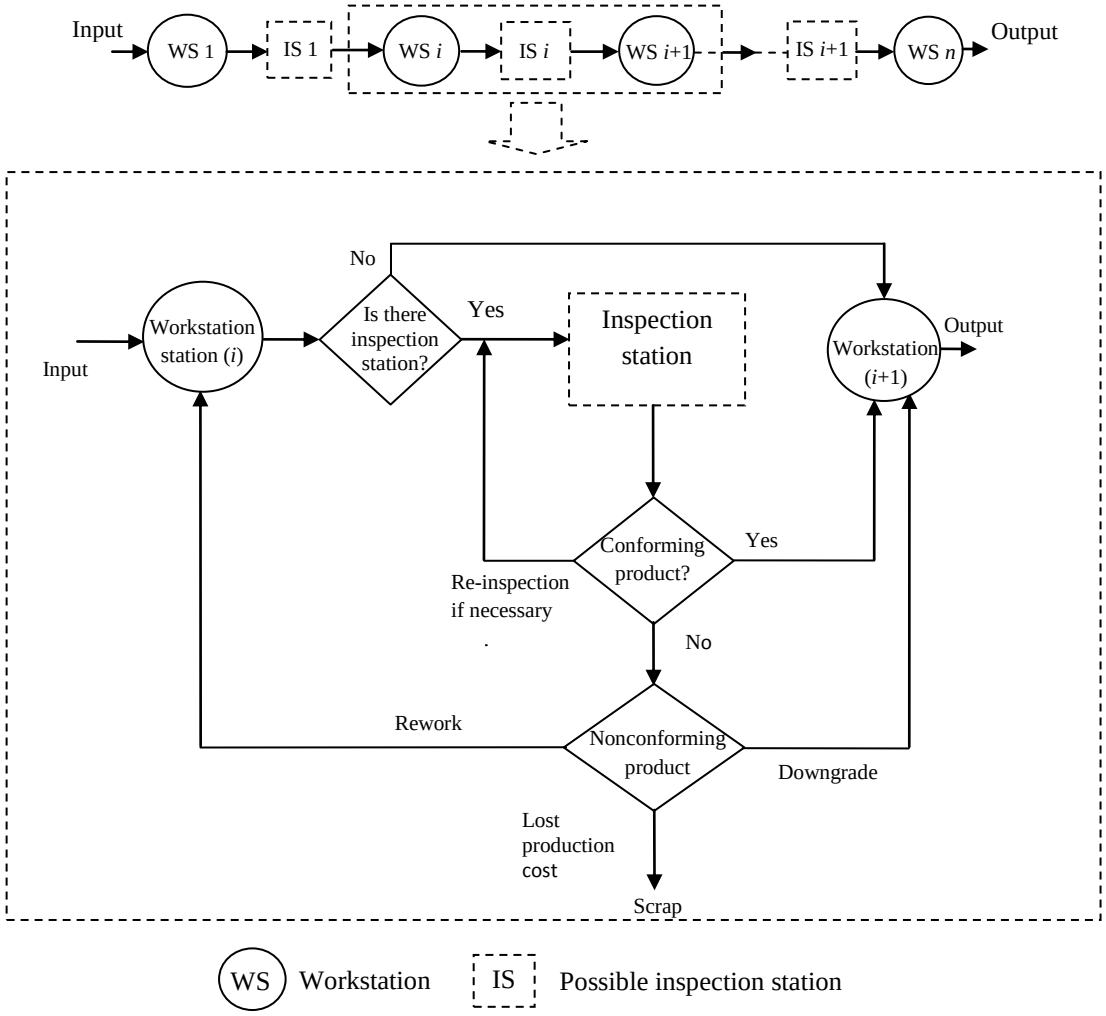


Figure 1.1: Inspection allocation problem in multistage manufacturing processes

In a serial multistage manufacturing process, as the problem size increases, so the number of inspection stations allocation possibilities increases exponentially in search space size. For example, in a serial multistage manufacturing processes consisting of n=24 processing workstations, there are 2^{24}= 16,777,216 possible combinations for allocating inspection points, and the complete enumeration method becomes impractical. This rapid growth in problem search space size is illustrated in Figure 1.2. It is possible that not all locations of inspection stations are economically equivalent; more likely, given differentials in cost structures and process characteristics, some combinations of inspection places may prove to be economically preferable to others.

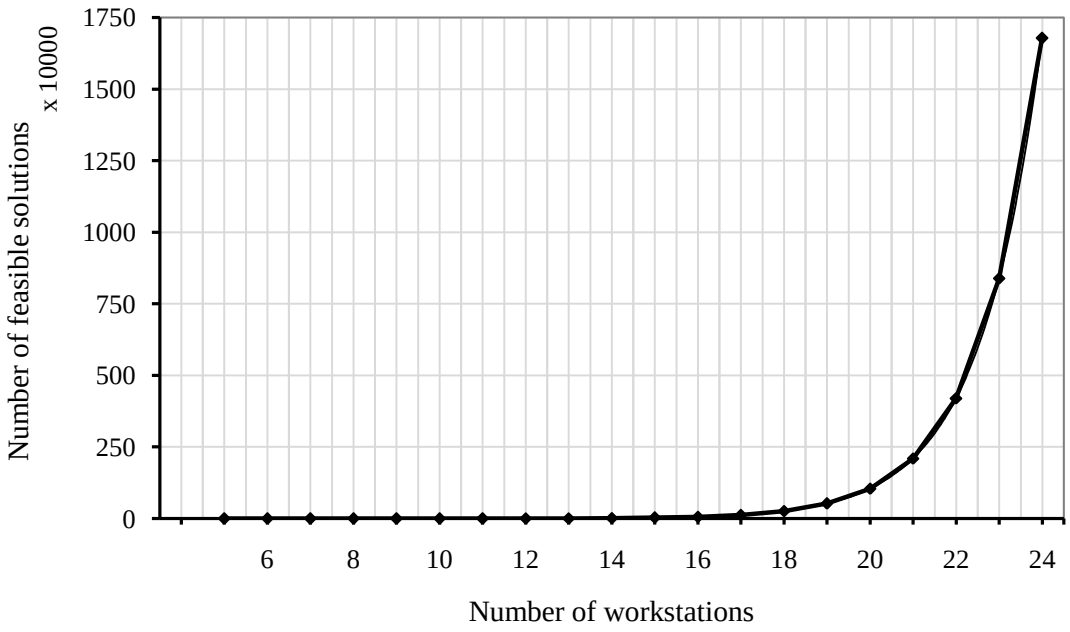


Figure 1.2: Growth of allocation of inspection stations problem search space. The horizontal axis is the number of workstations in an allocation problem. The vertical axis is the number of feasible solutions that have to be considered. The figure shows an exponential growth in search space size against problem size.

The problem therefore is where to locate limited number of inspection stations throughout the process to strike a balance between minimising the total cost, by capturing defective items and maintaining the required quality of the product.

1.2 Scope and objective of the research

The overall objective of this research is to develop a methodology for the allocation of inspection stations (AOIS) problem in serial multistage manufacturing processes. The methodology takes into account the constraints on inspection resources, in terms of a limited number of inspection stations. As a result the total manufacturing cost of a product can be reduced while maintaining the quality of the product. Given the scope of the research, the main areas of work completed are:

- To develop general cost models for the AOIS problem in serial multistage manufacturing processes. The developed models will be studied with the assumption of a limited budget, and that allocating limited inspection stations reduces the total manufacturing cost.
- To characterise the features of the AOIS problem. To identify the best possible approach method, different optimisation methods were investigated in chapter 4, leading to the ant colony optimisation technique. In chapter 5, different ant colony versions were studied, leading to the MMAS algorithm being proposed to tackle the AOIS problem.
- To develop a MMAS algorithm to solve the AOIS problem. Heuristic information for the MMAS is created. To improve the performance of the MMAS algorithm, local search methods for the MMAS algorithm are developed. Also, the most influential parameter values for the MMAS algorithm for the AOIS problem are well tuned.
- In order to evaluate the developed MMAS algorithm a genetic algorithm, simulated annealing method, particle swarm method, a pure random search algorithm and rules of thumb, are developed. A complete enumeration method has been carefully designed and used as benchmark to evaluate the developed algorithms.
- To develop a case study of serial multistage manufacturing processes to apply the MMAS algorithm. Experiments conducted to examine the results obtained from the MMAS algorithm. The results obtained will be discussed and analysed.
- To select two case studies for serial multistage manufacturing processes from the literature review, and apply the MMAS algorithm to these selected case studies. Experiments conducted to test the results obtained from the MMAS algorithm in comparison with the results of the selected case studies. The results obtained will be discussed and analysed.
- To select a real world case study for serial multistage manufacturing processes to apply the MMAS algorithm. Experiments conducted to examine the results obtained from the MMAS algorithm. The results obtained will be discussed and analysed.

1.3 Thesis structure

The thesis is organised as follows: Chapter 2 presents a literature survey and classifies the features of the surveyed models in terms of techniques applied, constraints used, type of multistage manufacturing system and type of costs considered by these models. In Chapter 3, a serial multistage manufacturing process is formulated. The model is developed under assumption that there are limited inspection stations available. The relation between quality and cost is described as well in this Chapter. Comparing computational time against number of workstations when using CEM is discussed. Computational complexity of combinatorial optimisation problems is presented. Chapter 4 describes and investigates different optimisation methods to identify the appropriate approach method for the AOIS problem. Chapter 5 presents ant colony optimisation in order to provide the necessary background on how ant colony optimisation algorithms are put together. The background of the field of ant algorithms has been described. Fitness landscape for AOIS problem is discussed. The fitness distance correlation indicates that the max-min ant system algorithm is well suited to the AOIS problem. In Chapter 6, a new approach of max-min ant system is developed to solve the AOIS problem. The max-min ant system algorithm will be used in combination with local search. Chapter 7 describes different case studies from the literature review. The appropriate case studies will be selected to test the proposed max-min ant system algorithm. Chapter 8 will introduce tuning the most influential parameters that one has to specify to instantiate the MMAS algorithm and the other relevant algorithms. In Chapter 9, behaviour of MMAS algorithm and sensitivity of control parameters for the AOIS problem are studied. In Chapter 10, experimental results and discussions are presented. In Chapter 11, the conclusions are drawn and possible directions to extend the research are discussed.

Chapter 2
Literature review

The AOIS in multistage manufacturing systems has received considerable attention from various researchers over the past decades. In this chapter, a relevant literature review in the area of AOIS in multistage manufacturing processes is conducted and analysed. This review covers the existing approaches and provides a classification of the models proposed in terms of system configuration, constraints, inspection errors, internal and external failure costs, inspection costs, manufacturing costs and the solution techniques applied. These characteristics will be described and summarised in tables in the same sequence as they appear here. The aim is to identify promising directions of research and to bridge a gap in the literature review.

2.1 Modelling characteristics

This literature review studies and investigates the different models proposed and solution techniques applied to solve the AOIS problem. This survey covered 51 published papers in the area of AOIS problems. A summary of each paper is represented by the first author's name, followed by a two-digit publication year. The order of articles is organised in chronological order by the year the paper was published. The summary has been produced which includes the main characteristics considered and solution techniques used in the current research publications. The solution methods are divided into exact and metaheuristic methods. In the following subsections, the main characteristics will be described first, followed by a short description for solution techniques which were used in the literature review.

2.1.1 System configuration

There are three types of production configuration, such as: (i) serial systems; (ii) assembly systems; and (iii) non-serial systems. In a serial production system, the raw materials pass through a sequence of processing workstations to produce the final product. Each stage of the manufacturing process receives a batch or flow of similar processing items as input, which may contain some mix of conforming and non-conforming units. While in an assembly system, at a certain stage, the product may be assembled with products from other processing lines. A system that is neither serial nor assembly falls into the category of a non-serial system (Mandroli et al., 2006).

2.1.2 Constraints

Constraints in the AOIS problem are related mostly to the characteristics of the manufacturing system, such as the structure of the system, the type of defect, and the type of inspection. However, additional constraints may be also imposed when solving the AOIS problem. These constraints may include a limited number of inspection stations, average outgoing quality limit (AOQL) and the rate of inspection (Raz, 1986).

2.1.3 Inspection errors

Two types of error may be generated by the inspection procedure: type-I error and type-II error. A type-I error refers to rejection of good items, and is also known as ‘producer risk’. A type-II error refers to the acceptance of non-conformance item and forwards it for further processing, and is also known as ‘consumer risk’. A type-II error is usually has great impact on the manufacturing process than type-I (Montgomery, 1997). This is because when non-conforming items reach to the customer unnecessarily greater costs will incur.

2.1.4 Internal and external failure cost

Internal failure costs incur inside the company, such as the cost of reworking, scrapping and downgrading. External failure costs incur after the products are shipped to customers for example, repairing, quality loss and replacement. Usually external failure cost items are represented as aggregated (penalty cost), and is usually associated with the final production of undetected non-conforming items that reach the customer (Montgomery, 1997).

2.1.5 Inspection cost

The inspection cost is a sum of the fixed cost and the variable cost. The fixed cost is a sum of the costs connected with test-equipment, installation and set-up. The variable cost is the total number of conforming parts and the number of defective parts produced at an inspection station, multiplied by the unit inspection cost (Mandroli et al., 2006).

2.1.6 Manufacturing cost

The manufacturing cost is the sum of the costs of all resources consumed in the process of making a product. The manufacturing cost is classified into three categories: direct material cost; direct labour cost; and overhead cost. Direct material cost results from adding a value to raw materials by applying a chain of operations to a product. Direct labour cost is the cost of workers that can be easily identified with the unit of production. Overhead cost includes all charges that provide support to manufacturing (Ostwald and McLaren, 2004).

In summary, the main characteristics of the AOIS problem were described. These characteristics were used by the developed models in the relevant literature. As will be shown in next subsections not all the surveyed papers were addressing all these characteristics. This is to allow a tractable formulation model and solution.

2.2 Exact methods

The concept of exact algorithms is based simply on enumerating the full solution space. Exact methods are guaranteed to find an optimal solution for discrete problems in bounded time such as is the case for many combinatorial problems (Ridge, 2007). However, in the worst case, as the problem becomes more complex, the time needed to solve it may grow exponentially. The following subsections describe exact methods which were used by the literature review, followed by a summary of each paper.

2.2.1 Integer programming (IP)

An integer program is a linear program in which all variables must be integers. The aim of integer programming is to find optimal decisions for problems where the decisions may only take a certain number of finite values (Hillier and Lieberman, 2010).

Park et al. (1988) used an integer programming technique to determine the location of inspection points in a flexible manufacturing cell. The objective was to minimise the expected total manufacturing costs. A numerical example was solved using the integer programming technique, which showed that inspection points depend on the reliability of the processing machine and the processing time of the following machine. Table 2.1 presents a summary of the classifications and characteristics of the studied model using IP method for the previously surveyed paper. It should be noted that (Yes) means that the characteristic is considered by the paper and (-) means that this characteristic is not considered.

Table 2.1: Classification the main characteristics for the studied model used IP method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Park (88)	Serial	Limited inspection stations	4	Free of error	Yes	–	Yes	Yes	IP

WS: workstation, IE: inspection error, IFC: internal failure cost, EFC: external failure cost, IC: inspection cost and MC: manufacturing cost.

Summary

For a real manufacturing system, the computations required by IP will rise significantly as the number of workstations increases, and its capability in terms of solving complex problems is limited (Liang and Smith, 2004).

2.2.2 Linear programming (LP)

Linear programming is a technique for the optimisation models in which the objective and constraints functions are strictly linear (Hamdy, 2003). Yum and McDowell (1987) developed a model using LP for solving inspection allocation problems for serial systems. They noted that none of the previous models in their literature review included a combination of reworking, replacement, repair and scrapping. They developed a model able to include any of these combinations. The objective of their work was to minimise the expected total cost. It was found that the optimal inspection policy is dependent on whether a production or a material requirement is used. Table 2.2 presents a summary of the classifications and characteristics of the model using LP for the previously surveyed paper.

Table 2.2: Classification the main characteristics for the studied model used LP method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Yum (87)	Serial	–	10	I and II	Yes	–	Yes	Yes	LP

I and II: false rejection of good units and false acceptance of defectives, respectively.

Summary

Mandroli et al. (2006) introduced a survey on the AOIS problem, and pointed out that the processing time required by LP will rise significantly as the number of workstations increases.

2.2.3 Non-linear programming (NLP)

Non-linear programming is the process of solving a system of equalities and inequalities, collectively termed ‘constraints’, over a set of unknown real variables. The objective function

is maximised or minimised, where some of the constraints or the objective function are non-linear (Hillier and Lieberman, 2010).

Numerous models in the existing literature review have been formulated the inspection allocation problem using the NLP technique. Ballou and Pazer (1982) developed a computer program to perform a 'what-if' simulation analysis of the serial systems with inspection errors. In a series of experiments, the authors found that inspection error rates have a major impact on cost. Type I errors (rejection of conforming units) were found to have a greater impact than type II errors (acceptance of non-conforming units). This contradicts the description in section 2.1.3. They interpreted that many real world systems continue to put pressure on the inspector to avoid type II errors which may be detected further down the line while failing to properly audit type I errors which may exist among the items discarded at the inspection station.

The original model of Ballou and Pazer (1982) was extended by the same authors in (1985), to analyse the relative merits of enhanced inspection and process improvement. The framework for their work assumes a multistage serial production system, with the possibility of end point and intermediate inspection. The inspection configuration chosen is one which minimises cost per good unit delivered to the customer and, accordingly, incorporates various cost trade-offs.

Tayi and Ballou (1988) noted that the traditional inspection procedures which incur costs are only used to identify and remove defective units. They proposed a model that considered both inspection and reprocessing activities. However, they assumed that the inspection configuration is given and fixed, and obtained a simple formula for determining the optimal initial lot size and the reprocessing batch size, which minimised the total system costs.

Barad (1990) described a break-even approach for performing inspection in a multistage production process. This paper was assumed that when inspection does take place, 100% of the processed product at that stage is inspected. One of the variables used to decide whether to inspect is the quality level at some points in the manufacturing process. Barad suggested allocating most of the inspecting resources to stages with a relatively high proportion of non-conforming product.

Jewkes (1995) noted that previous models of optimal inspection allocation in their review did not consider the case when a repair was carried out on-line. This paper was modelled inspection policies for a single stage manufacturing system as queues with two phases of service (processing and inspection), in which items can be inspected or repaired as necessary. Several examples were given to illustrate the process of finding the optimal effort.

Narahari and Khan (1996) proposed an approximate analytical technique based on mean value analysis for a non-serial manufacturing system. The aim was to predict the mean cycle time and throughput rate of such models under different inspection strategies. The proposed method has been validated using a simulation technique.

Lee and Unnikrishnan (1998) combined the inspection allocation problem with multiple inspection stations, in a scenario controlled by the inspection time constraint. Owing to the complexity of the problem, they developed three heuristic methods. The optimal solution was determined by the inspection plan that minimises the total cost per conforming item which exits the system. They found that the results of the heuristic methods were close to the optimised solution.

Shiau (2002) noted that the inspection error still needs to be considered for solving the allocation problem in a multistage manufacturing system. This paper has introduced an inspection error model and as a result of the complexity of the problem, two heuristic

methods were developed. The results show that solutions obtained by the heuristic methods were close to the optimal solution.

Kogan and Raz (2002) applied optimal control theory to determine the best AOIS in a serial system, to minimise the sum of the inspection cost. They assumed that the defect detection rate and the cost are linearly proportional with inspection. They obtained the optimal sequencing of inspection activities at a point in time, as well as optimal timing of switching between inspection points.

Emmons and Rabinowitz (2002) dealt with an inspection system for detecting malfunctioning operations in a multistage production system. The authors proposed a heuristic procedure for inspection assignment and scheduling that enables the prediction of the system performance under any inspection capacity. The main contribution of their paper was a theoretical foundation for the further development of models and solution procedures for more realistic problems. They demonstrated that the solutions obtained by the heuristic method are close to the optimal solution.

Shiau (2003a) noted that the inspection error needs to be considered even when applying the same inspection station to monitor various workstations that have different manufacturing capabilities. Based on the limited inspection resource constraint, Shiau developed a unit cost model and introduced two heuristic solution methods. It was found that the heuristic methods produce solutions near to the optimal solution, with less processing time comparing with CEM. Shiau (2003b) extended his previous model to include external costs, and proposed a heuristic method to solve the allocation problem. The results obtained were very similar to the original work.

Hadjinicola and Soteriou (2003) developed a mathematical model for a multistage production system. The authors noted that previous literature did not adequately examine the impact of

changes in the yield of each production stage on the total cost, from defects observed at each production stage. They developed a generalised model concerned with allocating a limited inspection budget to the different production stages. The aim was to improve the yield of the production stages and to minimise the cost. They concluded that the optimal budget allocation, leading to reduce in the annual expected cost, resulted from defects observed at all production stages.

Work on allocating inspection stations has been done by Rau and Chu (2005), which considered non-serial production systems. Owing to their complexity, a heuristic solution method was developed and proved to have much less calculation time, even when the number of workstations increases. The results of work Rau and Chu (2005) were used by Rau et al. (2005) to develop a mathematical model to find an optimal solution for allocating inspection stations in non-serial production systems. They used similar assumptions and considerations for the treatment of detected non-conforming items, as in the original model. To approach the complexity of the problem, a heuristic method was developed and the results obtained were very similar to the previous work.

Summary

The non-linear programming technique is the most popular method in the literature review, and was used by 15 papers, or 29% of the total. This is because of the nature of the inspection allocation problem in which some of the decision variables can only have integer values; for example, whether or not to inspect at the workstation. Moeini and Afshar (2009) explained that, for an actual manufacturing system, the computations required by NLP will escalate considerably as the number of workstations increases. However, its capability is limited in terms of solving a large-scale problem. In addition, they described that, within the past decade, many researchers have shifted their focus on optimisation problems from traditional optimisation techniques, based on linear and non-linear programming, to the implementation

of metaheuristic methods. Table 2.3 presents a summary of the classifications and characteristics of the studied models using NLP for the previously surveyed papers.

Table 2.3: Classification the main characteristics for the studied models used NLP method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Ballou (82)	Serial	–	3	I and II	Yes	Yes	Yes	–	NLP
Ballou (85)	Serial	_	3	I and II	–	Yes	Yes	Yes	NLP
Tayi (88)	Serial	–	5	Free of error	–	Yes	Yes	Yes	NLP
Barad (90)	Serial	–	8	Free of error	Yes	–	Yes	Yes	NLP
Jewkes (95)	Serial	–	_	Free of error	Yes	Yes	Yes	–	NLP
Narahari (96)	Non-serial	–	4	Free of error	Yes	–	_	–	NLP
Lee (98)	Serial	Inspection time	_	I and II	Yes	_	Yes	Yes	NLP
Shiau (02)	Serial	Limited inspection stations	7	I and II	Yes	Yes	Yes	Yes	NLP
Kogan (02)	Serial	–	6	Free of error	Yes	Yes	Yes	–	NLP
Emmons(02)	Non-serial	–	9	Free of error	–	–	Yes	Yes	NLP
Shiau (03a)	Serial	Limited inspection stations	7	I and II	Yes	_	Yes	Yes	NLP
Shiau (03b)	Serial	Limited inspection stations	5	I and II	Yes	Yes	Yes	Yes	NLP
Hadjinicola (03)	Assembly	–	3	Free of error	Yes	–	Yes	Yes	NLP
Rau (05)	Serial	–	16	I and II	Yes	Yes	Yes	–	NLP
Rau et al.(05)	Non-serial	Limited inspection stations	16	I and II	Yes	Yes	Yes	–	NLP

2.2.4 Branch and Bound (B&B)

The first Branch & Bound algorithm was developed in 1960 by A. Land and G. Doig for a general mixed and pure integer linear programming (ILP) problem (Hamdy, 2003). B&B is a general algorithm for finding optimal solutions to various optimisation problems, especially in discrete and combinatorial optimisation. Dorigo and Stutzle (2004) defined combinatorial optimisation problems as: involve finding values for discrete variables such that the optimal

solution with respect to a given objective function is found. The basic concept underlying the B&B technique is to divide and conquer. Since the original problem is too difficult to be solved directly, it is divided into smaller and smaller sub-problems until these sub-problems can be conquered. The division (branching) is done by partitioning the entire set of feasible solutions into smaller and smaller subsets. The conquering is done partially by bounding how good the best solution in the subset can be, and then discarding the subset if its bound indicates that it cannot possibly contain an optimal solution for the original problem (Hillier and Lieberman, 2010).

A few researchers used the B&B technique to solve the AOIS problem. Raz and Bricker (1987) used the branch and bound approach to tackle the problem of sequencing imperfect inspection operations. The aim was to find the variable inspection policy that minimises the total inspection cost. The developed model was subject to constraints type I and type II inspection errors. The authors found that the solution obtained by the proposed method was very close to the optimal solution.

Raz and Kaspi (1991) examined the sequencing and location issues for multiple inspection operations in serial production workstations. A single product type flowing in a fixed linear sequence was considered. The objective was to minimise the total expected manufacturing cost. They found that the solution obtained by the proposed method was close to the optimal solution. Table 2.4 shows an abstract of the classifications and characteristics of the models using B&B for the previously surveyed papers.

Table 2.4: Classification the main characteristics for the studied models used B&B method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Raz (87)	Assembly	I and II	5	I and II	Yes	–	Yes	–	B&B
Raz (91)	Serial	–	10	I and II	Yes	Yes	Yes	–	B&B

Summary

One of the drawbacks of the branch and bound method is that it requires a good initial upper bound, and an efficient way to calculate lower bounds for the various partitions (Raz, 1986). In addition, it is not always easy to find effective lower bounds (McCallum, 2005).

2.2.5 Dynamic programming (DP)

Dynamic programming was formalised in the early 1950s by mathematician Richard Bellman. DP is a recursive method that determines the optimum solution to an *n*-variable problem by decomposing it into *n* stages, with each stage constituting a single variable sub-problem. The computational advantages are that DP optimises single variable sub-problems (Hamdy, 2003).

A number of mathematical models have been developed to determine the optimal location of inspection stations in multistage production systems, using the dynamic programming technique. Lindsay and Bishop (1964) were the first to develop a model for determining an optimal inspection policy with the lowest total cost for serial production. The inspection is assumed to be perfect (no inspection error) and the inspection at one stage is independent of the next. It was found that DP allows the determination of a minimum cost under the added assumptions of maintaining a specified quality level, or when the cost associated with outgoing defective material is linear.

White (1965) also researched this area and showed that, with replacement of defectives, the optimal plan would be characterised by 0 or 100% inspection, and could be solved by a dynamic program. The results of the work of Lindsay and Bishop (1964) are used by Pruzan and Jackson (1967) to develop an adaptive model in which the optimal inspection policy at a location depended on the previous inspection history.

White (1969) presented two shortest route models for determining where to allocate inspection points on a serial production line. In this paper, both repairable and non-repairable defectives are considered. Hurst (1973) was the first to propose a model that considered two types of inspection errors: acceptance of non-conforming units (type I); and rejection of conforming units (type II). The production system was assumed to be serial with only one inspection operation possible after each processing stage, and units perceived to be non-conforming removed from the production flow.

In addition, Enrick (1975) and Hsu (1984) have applied dynamic programming to find the optimal location of inspection stations in serial systems. They concluded that dynamic programming was an effective technique for determining the inspection policies sequencing for a limited number of production stages.

Peters and Williams (1984) investigated the performance of five heuristics rules of thumb in a serial production system. The results indicated that a variety of economic and operating factors influenced the applicability of each of the five inspection location heuristics examined.

An inspection planning model was developed by Gunter (1985), for an assembly process free of error. The results shown that if defective items are removed from the line, then the production volume in the model will shrink as a result of inspection, in order to meet the demand the inspections decisions should be considered the production rates.

Raghavachari and Tayi (1991) developed a model as a shortest path algorithm in serial production systems to minimise the total cost. In their model, the determination of optimal initial lot size, inspection configuration and reprocessing decisions are considered simultaneously. They concluded that the developed model is flexible and versatile enough to be applied to the manufacturing environment with different characteristics.

Chengalur et al. (1992) extended the model of Ballou and Pazer (1985) by using a dynamic model with uncertainty in the quality of incoming raw material. They assumed 100% screening if inspection was performed at any stage. Also any defective item delivered to the customer was assigned as penalty cost. Their conclusion was that using a dynamic procedure led to cost-saving, even when unsure about the quality of the raw materials.

Bai and Yun (1996) investigated the problem in which a product consists of many identical components. In their model, only a limited number of (automatic) inspection machines are available, and the rate of production was constrained by the rate of inspection. The inspection level was defined as the percentage of components to be inspected. An inspection cost model was developed to obtain the best location for inspection points and the optimal inspection level. They found that the proposed heuristic algorithm combined with DP provides the optimal solution when the problem is small. However, as the problem increases, the heuristic algorithm provides a solution close to the optimal solution in less time comparing to the CEM.

An unreliable serial production system with known failure probabilities at each workstation was studied by Penn and Raviv (2007). The dynamic programming technique and a branch and bound method were used, to solve the problem of determining optimal quality control station configuration within the assembly line. The contribution of the model was incorporation of holding costs. Optimal quality control stations were found to reduce the load on the bottleneck stations, as well as the work in-process on the stations that followed them.

Table 2.5 presents a summary of the classifications and characteristics of the models using DP for the previously surveyed papers.

Summary

Dynamic programming technique is used in 13 papers, accounting for 25% of the papers surveyed. It is the second most common technique used, mostly used in the period from 1964

to 1985. In fact, the use of recursion in the DP algorithm has both advantages and disadvantages: the main advantage is usually simplicity; the major disadvantage is the rapid rise in computational requirements, as the number of decision variables to be optimised is increased. This has led to various algorithms being developed that limit the number of stages, states and decision variables combinations to be evaluated. In addition, Shiau (2002, 2007), Lee and Unnikrishnan (1998) and Rau and Chu (2005) have pointed out that the DP approach becomes impractical as the set of possible combinations grows exponentially. That is why many kinds of metaheuristic methods, such as simulated annealing, Tabu search and genetic algorithm are often used to reach a satisfactory solution, even though it may not be the optimal one.

Table 2.5: Classification the main characteristics for the studied models used DP method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Lindsay (64)	Serial	AOQL	9	Free of error	Yes	–	Yes	–	DP
White (65)	Serial	–	6	Free of error	Yes	–	Yes	–	DP
Pruzan (67)	Serial	–	5	Free of error	Yes	Yes	Yes	Yes	DP
White (69)	Serial	_	5	Free of error	Yes	Yes	Yes	Yes	DP
Hurst(73)	Serial	_	9	I and II	Yes	_	Yes	_	DP
Enrick (75)	Serial	–	_	I and II	Yes	–	Yes	Yes	DP
Hsu (84)	Serial	–	4	Free of error	Yes	–	Yes	Yes	DP
Peters (84)	Serial	–	13	Free of error	Yes	Yes	Yes	Yes	DP
Gunter (85)	Assembly	_	9	Free of error	Yes	–	Yes	–	DP
Raghavach-ari (91)	Serial	–	5	Free of error	Yes	Yes	Yes	Yes	DP
Chengalur (92)	Serial	Limited inspection	3	I and II	Yes	Yes	Yes	Yes	DP
Bai (96)	Serial	Limited inspection	10	I and II	Yes	Yes	Yes	_	DP
Penn (07)	Assembly	_	8	Free of error	–	Yes	Yes	Yes	DP

AOQL: average of outgoing quality level.

In summary, many models in the area of the AOIS problem have been reviewed. These models used traditional techniques to tackle the AOIS problem. However, the capability of these techniques, in terms of computational time required to obtain good solutions increases as the number of workstations increased significantly (Van Volsem and Neirynck, 2009).

2.3 Metaheuristic methods

A metaheuristic is defined to be a general heuristic method which is used to guide an underlying local search algorithm toward promising regions of the search space containing high quality solutions (Osman and Laporte, 1996). The following subsections describe metaheuristic methods, which were employed in the literature review, followed by a summary of each paper.

2.3.1 Simulated annealing (SA)

Simulated annealing simulates the thermodynamic behaviour of atoms suspended in a hot metallic liquid which is being cooled down over a period of time. This process is known as 'annealing'. SA is an intelligent approach designed to tackle complex problems within a reasonable computation time. The idea in SA, similar to iterative improvement, is to create some random perturbation, such as moving a molecule to a new location, and then the resulting change in energy, ΔE is evaluated. If the energy is decreased, $\Delta E<0$, the new configuration has less energy and is accepted as the initial point for the next move. However, if the energy is increased, $\Delta E >0$, the new, higher energy configuration is possibly acceptable with some probability (Eglese, 1990). A more detailed description of the mathematical model will follow in section 4.2.1.

A few models in the existing literature review were developed using SA, to determine the optimal location of inspection stations in multistage production systems. Chen and Thornton (1999) described how to allocate inspection location quantitatively, in a complex assembly

system, by using a combination of modelling, simulation and simulated annealing. The aim was to remove the greatest variation of a product at the lowest cost. They demonstrated that an optimal inspection plan can be selected quantitatively using their model.

Kakade et al. (2004) extended the model of Bai and Yun (1996) to account for the different quality characteristics after each processing station and the different inspection times required to inspect each component at each stage, along a serial multistage manufacturing system. They assumed that the rate of production was constrained by the rate of inspection. It was found that for small problems SA generates solution close to optimal solutions.

Table 2.6 presents a summary of the classifications and characteristics of the models using SA method for the previously surveyed papers.

Table 2.6: Classification the main characteristics for the studied models used SA method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Chen (99)	Assembly	–	_	Free of error	Yes	–	Yes	Yes	SA
Kakade (04)	Serial	Rate of inspection	2	Free of error	Yes	Yes	Yes	–	SA

Summary

The simulated annealing method has been highly successful in many applications and a number of variant algorithms. However, a major disadvantage of the technique is determining the ‘cooling schedule’. For example, deciding what is a sufficient amount of iterations at each temperature is difficult. In addition, determining the initial temperature is also difficult. Starting too high will waste computation time; starting too low will decrease the probability of finding a good quality solution given a hard enough optimisation problem (Ram et al., 1996).

2.3.2 Tabu search (TS)

Tabu search was proposed originally by Glover (1986) and has been subject to extensive studies, as well as applied to several optimisation problems with great success. Tabu search is equipped with a special mechanism to avoid being trapped in local optima. It uses a short-term memory to escape from local minima. TS typically uses a local search that, in each step, tries to make the best possible move from current solution *s*, to a neighbouring solution s', even if that move worsens the objective function value. In TS, to prevent the local search returning immediately to a previously visited solution and to avoid cycling, moves to recently visited solutions are forbidden (Glover, 1986). It uses a mechanism to forbid a return to a recently visited solution called 'Tabu list' (*tl*). TS uses some stopping criterion, such as a fixed number of iterations, a fixed amount of CPU time, or a fixed number of consecutive iterations without an improvement in the best objective function value. TS will be described in more detail in chapter 4.

Valenzuela et al. (2004) formulated an optimisation model for the allocation of paste-printing inspection efforts in Surface Mount Technology in a serial line. The aim was to maximise the expected total gain. To reduce the complexity, only one stage in the solder-paste printing process was considered. Their contribution was based on providing an optimisation model that considered explicitly the economic trade-off between product yield and inspection accuracy. They concluded that the heuristic approach provides a solution that can reduce the total expected cost by 15% when it is started from a random solution. Table 2.7 presents summary of the classifications and characteristics of the studied model using TS for the previously surveyed paper.

Summary

One of the drawbacks of using Tabu search is that it brings with it a number of parameters and design decisions, most of which are not simple to set, such as neighbourhood size and

Tabu list size (*tl*). This, therefore, extends the amount of experimentation required to optimise the hybrid algorithm. For example if *tl* is chosen too small, cycling may occur; if it is too large, the search path is too restricted, and high quality solutions may be missed. A good parameter setting for *tl* can only be found empirically and requires considerable fine tuning (Stützle, 1998b).

Table 2.7: Classification the main characteristics for the studied model used TS method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Valenzuela (04)	Assembly	–	2	Free of error	Yes	Yes	Yes	Yes	Tabu search

2.3.3 Genetic algorithm (GA)

Genetic algorithm is inspired by the observation of natural processes in the real world. John Holland invented the first genetic algorithm in the 1960s. He mimicked the insight he got by studying Darwin's theory of evolution, which can be summarised as:

- The traits found in the parents are passed on to their offspring during reproduction.
- New traits are produced by variations or mutations that are naturally present in all species.
- A process termed ‘natural selection’ chooses those individuals that are best adapted to the environment.
- Variations can accumulate and produce new species over long periods of time.

According to Darwin, natural selection can be reproduced as survival of the fittest. The characteristics of the fittest individuals, encoded in their genes, are passed on to their offspring and keep propagating into new generations (Gaertner, 2004). GA will be described in more detail in chapter 4.

A number of mathematical models have been developed to determine the optimal location of inspection stations in serial production systems using the Genetic Algorithm technique. Taneja and Viswanadham (1994) designed a genetic algorithm to determine the location of inspection stations for serial and non-serial multistage manufacturing systems. The aim was to minimise the expected total cost per unit produced. In their model, two constraints were considered: an accepted outgoing quality level; and limited number of inspection places. It was found that the GA algorithm reaches good solution as the number of workstation increases with a reasonable computational time.

Viswanadham et al. (1996) formulated inspection allocation models for a serial and a special non-serial multistage manufacturing system, with the aim of determining the number and location of inspection stations. As a result, the expected total cost per unit produced is minimised. The problem was approached using GA and SA. The conclusion using these techniques reduced computation time dramatically, compared to the exhaustive search, and yielded a good approximation to the optimal solution.

Langner et al. (2002) developed a genetic algorithm subject to inspection errors, for solving the multistage inspection problem under the assumption that all stages must receive partial rectifying inspection. They noted that previous models assumed that some manufacturing stages received full inspection, and the rest none. The objective of their work was to minimise the total cost. Examples of the optimisation of a multistage inspection problem were solved, and they concluded that the solution technique could handle multiple objectives and quality constraints effectively and linear and non-linear constraints trivially.

Shiau et al. (2007) developed a cost model to solve the manufacturing resource allocation problem by performing process planning and inspection planning concurrently, in a serial system. They concluded that GA saved time when compared with a complete enumeration method.

Van Volsem et al. (2007) suggested a fusion between an evolutionary algorithm (Genetic) and discrete event simulation, to optimise the inspection strategies for a multistage production system (MSPS). Their contribution was based on jointly optimising the number and location of inspection stations and inspection limits (specification interval). Their model was developed under the added assumptions that a limited number of inspection machines were available, and the rate of production was constrained by the rate of inspection. The objective of the work was to reduce the process variance of the final product at minimum cost. Their results confirmed the potential of the hybrid method for optimising quality inspection.

The work introduced by Galante and Passannanti (2007) focused on a job-shop system, equipped with inspection stations, in order to optimise both inspection allocation and operation scheduling. The authors noted that the interaction between the two problems has not been treated in a job-shop environment. They concluded that interactions between inspection point location and operation scheduling, led to solutions with considerably lower cost.

Sadegheih (2007) researched the area of the AOIS problem by developing two artificial intelligence techniques: genetic algorithm and simulated annealing. The aim was to find the optimal location of inspection stations to reduce the total cost in serial manufacturing systems. Sadegheih found that the performance of the GA is much better than the simulated annealing in terms of solution quality. Table 2.8 presents a summary of the classifications and characteristics of the studied models using GA for the previously surveyed papers.

Summary

A genetic algorithm is an established field and a great deal of work has been done in trying to apply it to a range of applications. The performance of the GAs depends on the rates of the parameters, such as the population size, crossover rate and mutation rate. Determining the

size of the population is a crucial factor. Choosing a population size that is too small increases the risk of converging prematurely to local minima, since the population does not have enough genetic material to cover the problem space sufficiently. On the other hand, a larger population has a greater chance of finding the global optimum at the expense of more CPU time.

Table 2.8: Classification the main characteristics for the studied models used GA

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Taneja (94)	Non-serial	AOQL and Limited	5	I and II	Yes	Yes	Yes	Yes	GA
Viswanadh-am et al. (96)	Non-serial	–	5-25	I and II	Yes	Yes	Yes	_	GA
Langner (02)	Serial	AOQL	6	I and II	Yes	–	Yes	Yes	GA
Shiau (07)	Serial	Limited inspection stations	5	I and II	Yes	Yes	Yes	Yes	GA
Van Volsem (07)	Serial	–	6	Free of error	Yes	Yes	Yes	–	GA
Galante (07)	Serial	–	10	I and II	Yes	Yes	Yes	Yes	GA
Sadegheih (07)	Serial	–	10	Free of error	Yes	–	Yes	–	GA

In summary, different techniques based on metaheuristics in the area of the AOIS problem have been reviewed. These techniques are capable of finding, good and sometimes optimal, solutions to large size problems, in a generally shorter computation time. However, the main drawback of these techniques is that they bring with a number of parameters, most of which are not simple to set. Abramson and Abela (1992) explained that another drawback of genetic algorithms require large number of response (fitness) function evaluations depending on the number of individuals and the number of generations. Therefore, genetic algorithms may take long time to evaluate the individuals. This agrees with what found in the studies introduced by Espinoza et al., (2005) and Kim (2010).

2.4 Markov decision process (MDP)

A Markov decision process is a controlled stochastic process that: (i) assumes that every process state depends only on the previous process state and not on the history of previous states (Markov assumption); and (ii) assigns costs (or rewards) to state transitions (Di Caro, 2004). In general, an MDP is a restricted state with a state-transition feedback function described by (*X*, *U*, *T*, *J*) where:

X: is a finite set of problem states, representing the environment;

U: is a finite set of actions;

T: defines the transition probability distribution $P(x_i|x_j, u_k)$ that describes the effect of actions on the world state; and

J: defines a cost model that describes costs associated with a state transition under some action.

In the existing literature review, a few models used the MDP technique to solve the allocation problem. Deliman and Feldman (1996) extended the model of Ballou and Pazer (1985) in order to consider modelling rework directly, as reprocessing activities in serial manufacturing systems. The objective was to determine the positions of inspection stations so that the expected per unit total cost of production was minimised. Their results showed that an optimal combination of inspection and process improvement can be identified and saves cost.

Jang and Shanthikumar (2002) presented stochastic models, by using the Markov decision process to solve the inspection allocation problem under the stationary allocation assumption. Optimal allocation and limited inspection capacity for multiple production processes were considered. The main purpose was to minimise the expected total discounted cost over a finite time prospect. They concluded that the model could be used as a simple tool to evaluate the relative importance of individual processes. Table 2.9 gives a summary of the

classifications and characteristics of the studied models using a MDP for the previously surveyed papers.

Table 2.9: Classification the main characteristics for the studied models used MDP method

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Deliman (96)	Serial	–	5	II	Yes	Yes	Yes	Yes	MDP
Jang (02)	Serial	Limited inspection stations	6	Free of error	Yes	–	Yes	Yes	MDP

Summary

The Markov decision process can be very effective in practice, but when the state set is too large, or the states are not accessible, they are virtually ruled out (Di Caro, 2004). Hauskrecht (2000) pointed out that a significant drawback of the MDP is that it is only applicable for solving relatively simple problems.

2.5 Simulation

A simulation of a system is the operation of a model of the system (Maria, 1997). A simulation model is used to observe a real system with cases; usually, this is impossible, too expensive or impractical to do in the system it represents (Carson and Maria, 1997). The aim of simulation is to collect pertinent information about the behaviour of the system with the passage of time. Simulation is not an optimisation technique, and is used to estimate the measures of performance of a modelled system. It is a statistical experiment, and hence its output must be interpreted by appropriate statistical tests. Two broad categories of simulations are discrete-event and continuous simulations (Hillier and Lieberman, 2010). A discrete-event simulation is one where changes in the state of the system occur instantaneously, at random points in time, as a result of the occurrence of discrete events. A continuous simulation is one where changes in the state of the system occur continuously

over time. Most applications of simulation in practice are discrete-event simulation (Hillier and Lieberman, 2010).

There are many studies in the literature review that solved AOIS problems by using the simulation technique. Saxena et al. (1990) evaluated four inspection heuristics on the basis of job completion under different operating conditions in serial production systems. The four heuristics considered were: (i) locate one inspection station before the station with the longest processing time and locate another at the end of the total process; (ii) locate one inspection station after the operation which is likely to generate a high proportion of defective items and locate another at the end of the whole process; (iii) locate one inspection station after each machine; and (iv) locate one inspection station at the end of the whole process. They found that inspection time was the most influential factor for the selection of a particular heuristic.

Gardner et al. (1995) demonstrated the impact of defective rates, inspection and defective removal strategies on the profitability of a manufacturing system in an assembly line. A number of specific manufacturing situations have been simulated, to study particular aspects of performance or costs. They confirmed that the maximum profit at a zero defect rate occurred when inspection and defective removal were minimised.

Shin et al. (1995) investigated strategic AOIS for a flow assembly line. The problem was formulated as a constrained bottleneck, shortest path model. They investigated the variations of four parameters (defect ratio, inspection time, repair time, and inspection error) on the strategic allocation. Their contribution introduced a hybrid method by combining constrained bottleneck shortest path algorithm and discrete-event simulation. The results show that the strategic allocation yielded the maximum throughput, by balancing the segment time at each workstation.

Lee and Chen (1996) also researched this area by investigating five heuristics rules to study the effects of inspection sequencing rules and part scheduling policies, in a flexible

manufacturing system (FMS). The conclusion was the selection of inspection plans that were found to have a significant impact on the FMS performance.

Siemiatkowski and Przybylski (2006) also investigated the inspection heuristics of process flow planning within a machining cell, with a coordinate measuring machine. They used similar assumptions and considerations of the previous model presented by Lee and Chen (1996). The results show that job-sequencing strategies relied heavily on the inspection plan introduced in the cell. Table 2.10 presents outline of the classifications and characteristics of the studied models using a simulation method for the previously surveyed papers.

Table 2.10: Classification the main characteristics for the studied models used simulation

Article	Characteristics								
	System configu-reation	Constraints	Number of WS	IE	Cost components				Solution technique
					IFC	EFC	IC	MC	
Saxena (90)	Serial	–	5	Free of error	Yes	–	Yes	Yes	Simulation
Gardner (95)	Assembly	–	7	I and II	Yes	–	Yes	Yes	Simulation
Shin (95)	Serial	Throughput	7	II	Yes	–	Yes	–	Simulation
Lee (96)	Serial	–	9	Free of error	–	–	Yes	Yes	Simulation
Siemiatkow-ski (06)	Serial	–	2	Free of error	Yes	–	Yes	Yes	Simulation

Summary

Simulation usually only provides statistical estimates rather than exact results, and often compares different solutions rather than creating an optimal one. However, creating optimal solutions using simulation needs a special software package (Hillier and Lieberman, 2010).

2.6 Cost models for serial multistage manufacturing processes

This research (AOIS problem) focuses on serial multistage manufacturing processes. The main characteristics included in each cost model reviewed are broken down in more detail as follows: constraints, number of workstations, inspection errors (type I and type II), internal failure cost (rework and scrap), external failure cost (repair and replacement), inspection cost (fixed and variable), manufacturing cost and objective function. It should be noted that these

characteristics are presented in more details than in the previous tables. The main characteristics of each cost model are summarised in Table 2.11 in the sequence in which they are published.

It can be seen from Table 2.11 that the models developed by Lindsay and Bishop (1964) and White (1965) only included scrap and inspection costs. However, Lindsay and Bishop (1964) added the assumption of AOQL in their developed cost model. The model developed by Pruzan and Jackson (1967) included scrap, inspection (fixed and variable) and manufacturing costs. The same cost components were considered in the model developed by White (1969), with the addition of a repair cost.

Hurst (1973) developed a cost model which only included the scrap cost and the variable inspection cost. The main contribution of the paper was the incorporation of inspection errors. Enrick (1975) introduced a cost model to consider the rework, manufacturing and variable inspection costs under the added assumption of inspection errors. However, Enrick did not test the developed cost model.

Ballou and Pazer (1982) limited their cost model to include inspection errors, and scrap, penalty and variable inspection costs. The contribution was based on analysing the impact of inspection errors on the inspection policy. Hsu (1984) presented a cost model to consider scrap, variable inspection and manufacturing cost components. The main contribution of the work was the introduction of a hybrid inspection system.

Peters and Williams (1984) developed a cost model to include all characteristics except inspection errors. The model was developed without any constraints. The main contribution was the identification of the relations between various costs and/or process characteristics, and a designated ‘operative condition’ that motivated the apparent rationale of five of these rules-of-thumb.

Table 2.11: Cost model characteristics for serial multistage manufacturing processes

Article	Characteristics										
	Constraints	Number of WS	IE	Cost components							Objective function
				IFC		EFC		IC		MC	
				Rework	Scrap	Repair	Repl.	Fixed	Variable		
Lindsay and Bishop (1964)	AOQL	9	–	_	Yes	_	_	_	Yes	–	Cost/input unit
White (1965)	–	6	–	_	Yes	_	_	_	Yes	–	Cost/input unit
Pruzan and Jackson (1967)	–	5	–	_	Yes	_	_	Yes	Yes	Yes	Cost/input unit
White (1969)	_	5	–	_	Yes	Yes	_	Yes	Yes	Yes	Cost/input unit
Eppen and Hurst (1974)	–	9	I and II	_	Yes	_	_	_	Yes	–	Cost/input unit
Enrick (1975)	–	_	I and II	Yes	_	_	_	_	Yes	Yes	Cost/input unit
Ballou and Pazer (1982)	–	3	I and II	_	Yes	Penalty		_	Yes	_	Cost/input unit
Hsu (1984)	–	4	–	_	Yes	_	_	_	Yes	Yes	Cost/input unit
Peters and Williams (1984)	–	13	–	Yes	Yes	Penalty		Yes	Yes	Yes	Cost/input unit
Ballou and Pazer (1985)	_	3	I and II	_	_	Penalty		_	Yes	Yes	Cost/input unit
Yum and McDowell (1987)	Limited inspection stations	10	I and II	Yes	Yes	_	_	–	Yes	Yes	Cost/input unit
Tayi and Ballou (1988)	_	5	–	Yes	Yes	Penalty		–	Yes	Yes	Cost/input unit
Park et al. (1988)	–	4	–	Yes	Yes	_	_	Yes	_	Yes	Cost/input unit
Barad (1990)	–	8	–	Yes	_	_	_	_	Yes	Yes	Cost/input unit
Raghavachari and Tayi (1991)	–	5	–	Yes	_	Penalty		Yes	Yes	Yes	Cost/input unit
Raz and Kaspi (1991)	_	10	I and II	Yes	_	Penalty		_	Yes	_	Cost/output unit

Repl.: replacement

Table 2.11: Cost model characteristics for serial multistage manufacturing processes (continued)

Article	Characteristics										
	Constraints	Number of WS	IE	Cost components							Objective function
				IFC		EFC		IC		MC	
				Rework	Scrap	Repair	Repl.	Fixed	Variable		
Chengalur et al. (1992)	Limited inspection stations	3	I and II	_	Yes	Penalty		Yes	_	_	Cost/input unit
Jewkes (1995)	–	_	–	Yes	_	_	_	_	Yes	_	Cost/input unit
Bai and Yun (1996)	Limited inspection stations	10	I and II	Yes	_	Penalty		_	Yes	Yes	Cost/input unit
Deliman and Feldman(1996)	–	5	II	Yes	Yes	Penalty		–	Yes	Yes	Cost/input unit
Lee and Unnikrishnan (1998)	Inspection time	_	I and II	Yes	Yes	Yes	_	Yes	_	_	Cost/input unit
Jang and Shanthikumar (2002)	Limited inspection stations	6	–	Yes	_	_	_	Yes	_	_	Cost/input unit
Langner et al. (2002)	AOQL	6	I and II	Yes	_	_	_	_	Yes	_	Cost/input unit
Shiau (2003a)	Limited inspection stations	7	I and II	Yes	Yes	_	_	_	Yes	Yes	Cost/input unit
Shiau (2003b)	Limited inspection stations	5	I and II	Yes	Yes	_	Yes	_	Yes	_	Cost/input unit
Rau and Chu (2005)	–	16	I and II	Yes	Yes	Penalty		_	Yes	Yes	Throughput
Shiau (2007)	Limited inspection stations	10	I and II	Yes	Yes	_	Yes	_	Yes	_	Cost/input unit
Sadegheih (2007)	_	10	_	_	Yes	_	_	_	Yes	_	Cost/input unit

Ballou and Pazer (1985) extended their original model (Ballou and Pazer, 1982) in order to analyse the cost quality implication of process improvement against inspection enhancement. The model was developed to include only inspection errors, and penalty, variable inspection and manufacturing costs. However, the cost model did not include any items of internal failure cost.

Yum and McDowell (1987) developed a cost model that was constrained by adding assumptions for a limited number of inspection stations. The developed model did not include external failure costs (repair and replacement) or fixed inspection costs. The main contribution of the work was the introduction of a model that could include any combination of internal failure costs. Tayi and Ballou (1988) presented a model that produced several potential features of production systems: production processes, quality control procedures, in-process inventory and reprocessing. The model was developed to include all cost components, except fixed inspection costs, under the added assumption of perfect inspection (no inspection errors).

Park et al. (1988) developed a cost model to consider internal failure, fixed inspection and manufacturing costs. The main contribution was the incorporation of machine operation reliability. Barad (1990) introduced a concept of a break-even quality level for a multistage production process, in which screening was only allocated within some stages, depending on the economic criteria. When screening did take place, 100% of the product processed in that stage was inspected. One of the variables used to decide whether or not to inspect was the quality level at some point in the manufacturing process. The developed model was limited to only including rework, variable inspection and manufacturing costs.

Raghavachari and Tayi (1991) extended the cost model of Tayi and Ballou (1988) to provide an integrated framework for simultaneously considering manufacturing, inspection and reprocessing activities. The model excluded inspection errors and scrap costs. Raz and Kaspi

(1991) examined the sequencing and location of inspection stations in serial production workstations. Their main contribution was in performing multiple inspection operations after each production stage. The model was developed to include rework, penalty, and variable inspection costs, with the added assumption of inspection errors. However, scrap, fixed inspection and manufacturing costs were excluded in the cost model developed.

Chengalur et al. (1992) extended the cost model of Ballou and Pazer (1985) to consider the quality of the incoming raw material, and consequently the optimal inspection strategy. The developed model was constrained by a limited number of inspection stations. The developed cost did not include rework, variable inspection or manufacturing costs. Jewkes (1995) examined a system with a stationary probability of producing defective items and focused on an optimal fraction of items to be inspected. The developed model was limited to only include rework and variable inspection costs.

Bai and Yun (1996) investigated the AOIS problem in which a product consists of many identical components. In their model, only a limited number of inspection machines were available, and the rate of production was constrained by the rate of inspection. The cost model was developed to include all cost components, except for scrap and variable inspection costs. Deliman and Feldman (1996) expanded the model of Ballou and Pazer (1985) to consider internal failure costs (rework and scrap) in serial manufacturing systems. However, the developed model did not include a fixed inspection cost or type I inspection error.

Lee and Unnikrishnan (1998) developed a cost model with multiple inspection stations, in a scenario controlled by the inspection time constraint. The time constraint was used to simplify the problem and to speed up the process of finding a good solution. However, the model excluded the variable inspection cost. Jang and Shanthikumar (2002) presented stochastic models to solve the inspection allocation problem. The main contributions of the paper were the characterisation of a threshold type optimal inspection control policy for a

constrained single process problem and the introduction of a dynamic disaggregate approach that efficiently solves a multiple process problem. The cost models were developed to only include rework, fixed inspection and manufacturing costs. The cost developed models were constrained by the limited number of inspection stations.

Langner et al. (2002) developed a cost model to find the number of defective items in each manufacturing stage as a result of this the total cost can be reduced. The cost model was developed to only include inspection errors, and rework, variable inspection and manufacturing costs. The cost model did not consider scrap cost and external failure cost. Shiau (2003a) developed a unit cost model based on a limited inspection resource constraint. The unit cost model did not consider external failure or fixed inspection costs. Shiau (2003b) extended his previous model to include external costs. However, the cost model did not consider manufacturing or fixed inspection costs.

Rau and Chu (2005) developed a cost model for optimally allocating inspection stations in serial production systems. The main contribution was the incorporation of two types of workstation: a workstation of attribute data and a workstation of variable data, in serial production systems. The cost model was developed to consider all cost components, except for the fixed inspection cost and inspection errors which were excluded.

Shiau (2007) developed a unit cost model to solve the allocation problem in an advanced manufacturing system (AMS). The main contribution was in finding solutions by concurrently performing process planning and inspection planning. Except for fixed inspection and manufacturing costs, all other cost components were included in the developed cost model.

Sadegheih (2007) researched the area of the AOIS problem by developing a cost model. The developed cost model was limited to only include scrap and variable inspection costs. The

contribution of the paper was in the presentation of an optimal design for a manufacturing system inspection station using genetic algorithms and simulated annealing techniques. Furthermore, a novel general effect of the mutation rate on the minimised objective value was presented.

2.7 Conclusion

This literature review surveyed research in the area of AOIS in multistage manufacturing processes. A number of techniques have been presented in the literature for the AOIS problem. The aim of almost all the reviewed inspection models was to determine whether inspection operations should be performed at some or in all processing workstations to minimise the total cost. In some cases, there were many possible locations for inspection activity; while in other situations it was concluded that an effective inspection should performed after certain workstation processes had been completed. It was found that most of the surveyed papers used a complete enumeration method (CEM) to measure the performance of the models developed. The advantage of the CEM is that it checks all possible combinations of inspection stations in the search space. As a result the optimal solution can then be identified. The most common approaches used by researchers were the NLP and DP techniques. However, the computation time required by these traditional techniques to obtain good solutions increases when the number of workstations significantly increases. Of all the metaheuristic methods used in the literature review, none of them used local search to improve the performance of their models. For example Shiau (2007), Galante (2007) and Kakade (2004) all used approximation methods, but did not use a local search method to improve the performance of their optimisation methods. In addition, it has been proven that in another area (not an AOIS problem) hybrid metaheuristics perform better than metaheuristics without local search (Duda, 2006). It was found that a vast majority of the papers surveyed do not consider sensitivity analysis, which only appeared in some papers, for example the

models developed by Yum and McDowell (1987) and Parak et al. (1988).This is probably because their models were developed using Integer Linear Programming, and hence it is easily included. The complexity of cost models and the fact that no two such cost models are alike, led researchers to the use simulation techniques when examining such systems.

Despite the product and the process in multistage manufacturing processes being discrete as in Petri nets, none of the surveyed models mentioned about this similarity. Petri nets are based on analyses of system behaviour. These analyses will lead to important insights into the behaviour of the modelled system. For complex Petri net models, discrete-event simulation is used to check the system properties. Petri nets are particularly useful for modelling systems with concurrent and asynchronous processing (Wang, 2007).

Any realistic model of the impact of inspection on a multistage manufacturing system should include the possibility of inspection errors. Other papers pointed out that as the production processes at each stage are generally stochastic in nature, deviations from product specifications occur, which, without intervention, will accumulate during the course of the production process. Only performing an inspection at the last stage would therefore result in a large number of faulty products and high rework and scrap costs. Regarding the rules of thumb which were used by some papers, it was concluded that its effective application depends on the consideration of a variety of cost and/or process factors unique to each heuristic. Therefore, the specific results obtained are dependent on the unique set of experimental conditions.

Regarding assembly and non-serial models, undetected defects are more difficult to determine in these systems than in serial lines. The difficulty arises due to assembly stages in which multiple serial lines join to form a single serial line. At such assembly stages, the number of detected defective output flow items of the assembly stage entering the assembly line stage depends on the proportion of defective items leaving all series lines to that

assembly stage. Optimal solutions to the problems can be found when the full enumeration of the search space is generated in reasonable time. The specific optimal solutions obtained are dependent on the unique set of experimental conditions for each problem. The vast majority of case studies in the literature review presented the best solutions determined by the methods developed in terms of average deviation from the optimal solution.

It is evident from the preceding review, particularly in section 2.6, that none of the cost models reviewed address all the characteristics of the general inspection allocation problem. Simplified assumptions were introduced at several points in those studies, in order to allow a tractable formulation model and solution. This is also because many studies were interested in developing new heuristic methods to approach the complexity of the AOIS problem (e.g. Barad, 1990, Lee and Unnikrishnan, 1998, Peters and Williams, 1984, Rau and Chu, 2005, Shiau, 2003a, b).The categories of linear and nonlinear variable inspection costs are largely exclusive. As a result they are the categories of the objective functions and the optimisation methods. The linear and nonlinear variable inspection costs cover all papers studied, with the four exceptions of Park et al. (1988), Chengalur et al. (1992), Lee and Unnikrishnan (1998) and Jang and Shanthikumar (2002) who only considered a fixed inspection cost and no variable inspection cost. All the cost models surveyed assumed that when an inspection is performed after the processing workstation, 100% inspection occurs. This assumption increases the cost of inspection and inspection time. Also, all the cost models surveyed used the total cost per input unit and the total cost per output unit as the objective functions. The difference between them is that in the first case the objective function is computed as the total cost divided by the number of input units, whereas in the second case the objective function is determined as the total cost divided by the number of output units. However, the customer is often determines the quality of an item with 100% accuracy. This review has demonstrated

the need to develop a general cost model that can handle the increased complexity of the problem.

In this thesis, the AOIS problem is expanded to include all the characteristics described in section 2.6. The cost model is developed under assumptions of inspection errors and the limited availability of inspection stations. In addition, the developed cost model determines the locations of inspection stations using sampling inspection strategy. Furthermore, the optimality is defined in terms of minimising costs per conforming output unit accepted by the customer. To do so, the general cost model is developed such that the number of conforming parts can be computed at each processing workstation. Introducing all these issues into the developed cost model significantly increases the level of complexity and gives an advantage to be very different from those in previous literature. The novel general cost model for the AOIS problem will be developed in the next chapter.

Chapter 3
General cost model formulation

It is evident that from the previous review that the various cost models did not address all the complexities possible characteristics of the general inspection allocation problem. Also, all the studied cost models in the previous literature were used full inspection plan if an inspection station is located after a workstation. In addition, all the surveyed cost models used the total cost per input unit and the total cost per output unit as the objective function. Furthermore, all previous cost models were represented external failure cost items as aggregate or only include one of them. In this chapter, a novel general cost model (GCM) in serial multistage manufacturing processes to include all the characteristics of the AOIS problem is developed. The GCM is also contributes to knowledge by determining the locations of inspection stations using sampling inspection strategy. Furthermore, the external failure costs are represented to be more complex to include all of its items (repair and replacement costs). Also, the optimality is defined in terms of minimising cost per conforming output unit accepted by the customer. This can be done by developing the general cost model such that the number of conforming parts can be computed at each processing workstation. Introducing all these issues into the GCM significantly increases the level of complexity and gives an advantage to the general cost model to be very different from those in previous literature. This general cost model provides management with information on the optimal number and placement of inspection stations for specific planned or existing serial manufacturing systems. It can also be used by management to explore various policy options, such as the cost implications of increasing the quality vs. the quantity of inspection stations.

The relation between quality and cost, computational complexity and computational time against number of workstations will also be described in this chapter.

3.1 A serial multistage manufacturing process

In a serial production system, the raw materials pass through a sequence of processing workstations to the final product. Each stage of the manufacturing process receives a batch or flow of similar processed items as input, which may contain a mix of conforming and non-conforming units. As an example of a serial multistage manufacturing process, consider the manufacturing of cylinder heads for engine blocks. The cylinder head sits above the cylinder block, as shown in Figure 3.1, and consists of a platform containing part of the combustion chamber that locates the valves and spark plugs. In this serial line process, the work-piece moves from one machine to another, to enable various operations to be performed on the part. The system consists of 15 processing workstations in a serial line for producing cylinder heads for engine blocks. This system is capable of producing 100 cylinder heads per hour. The various operations performed by the machines are: milling, drilling, reaming, boring, tapping, honing, washing, and gauging (Kalpakjian and Schmid, 2006). Another example of a serial multistage manufacturing process, which is considered to be a large-scale problem, is manufacturing engine valves, as shown in Figure 3.2. The engine valves are processed in 36 different processing operations on different machines, such as cutting, turning, press,

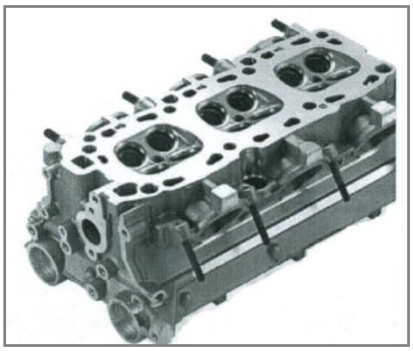


Figure 3.1: Cylinder head for engine block

Figure 3.2: Engine valves

welding, grinding, heat treatment and plating. These different machines are equipped with computer numerical controls (Engin, 2008). In addition, in real world problems there are more complex manufacturing processes such as manufacturing Trent 700 turbine blade (Ridley, 2008). This process is highly complex and can be broken down into 88 workstations, with each operation having around 12 steps. Due to the complexity of the process, Ridley (2008) summarised these operations into 12 discrete steps as follows: (1) diffusion bonding, (2) water jet cut profile and C scan, (3) hot process (twist), (4) hot process (creep forming), (5) hot process (super-plastic forming), (6) X-Ray, (7) Ghyll brow operations (chemical and out gassing), (8) CNC machining, (9) polishing, (10) coordinate measuring machine inspection, (11) finish processing (shot peening and super polishing) and (12) frequency and moment weight. Despite the processes using high precision machines, these operations still generate defective items, as shown in Table 3.1.

Table 3.1: Defective rates for Trent 700 turbine blade (Ridley, 2008)

Workstations	Number of defective items
1	70
2	0
3	2
4	1
5	293
6	0
7	4
8	24
9	49
10	0
11	8
12	0

3.2 Manufacturing system characteristics

The most common way used by Shiau (2003a, b), Mandroli et al. (2006) and Shiau (2007) to characterise a serial multistage manufacturing process are: (i) non-conforming products; (ii) repair of defects (iii) inspection type; (iv) inspection errors; (v) inspection time; and (vi) cost components.

3.2.1 Non-conforming products

Products may not conform because of the improper performance of a processing operation. The chance that a unit will become non-conforming at a given stage is referred to as the non-conforming processing rate for the stage, which may be constant or variable, and may alternate between an acceptable level and an out-of-control level (Raz, 1986). A given processing stage may cause a single type of non-conformity or multiple types. For an allocation problem, it may be assumed that each workstation has a specific probability of producing defective parts. Products considered to not conform are, subsequently, removed from the production flow and they may have some or no salvage value at all (Kakade, 2004 and Galante and Passannati, 2007). The salvage value represents the revenue generated by selling the rejected items as scrap or lower grade products.

3.2.2 Repair of defects

During the inspection, once an item does not conform to the specifications certain actions will be taken to repair, or simply scrap it. However, defective products, when allowed to pass through the production line, become costly to repair at a later stage of operation. The repair occurs only when the non-conformance of a product is greater than the specification limit, which is determined by a predefined quality requirement. In the absence of repair, the production volume in the production line will shrink as a result of inspection (Gunter, 1985). In real life, without repair larger lot sizes have to be introduced, in order to meet production plans and to avoid delivery delays.

3.2.3 Inspection type

If an inspection is performed after a particular workstation, it may belong to one of four categories: (i) a simple inspection by inspecting one single item once; (ii) a batch inspection; (iii) a repeated inspection by inspecting the same batch or a unit more than once; and (iv) a dynamic inspection to inspect units in a batch sequentially, and a decision of whether to reject

or accept the batch is made dynamically instead of at a fixed fraction (Mandroli et al., 2006). Repeated inspection is similar to dynamic inspection because they both inspect another inspection before reaching a decision. The difference between them is that the repeated inspection examines the same item or the same batch of units, whereas the dynamic inspection inspects a different item. It is also worth noting that this study on inspection strategies does not differentiate between inspections conducted by automated devices, human inspectors, or a mix of both (e.g., human inspection followed by an automatic inspection). This is because the actual inspection actions are usually modelled using a set of parameters that are independent of the actual inspection methods (such as type-I and type-II errors defined in section 3.2.4 and the number of repetitions). In reality, inspection may be performed due to legislation and regulations. This is a typical issue in industry what causes a lot of costs that could be reduced by optimising inspection stations through the processing workstations.

3.2.4 Inspection errors

During the inspection operation two types of error may be generated by the inspection procedure: type-I error and type-II error (Montgomery, 1997). Type-I and type-II errors can be summarised in following confusion matrix (Visa et al., 2011, Gluga et al., 2011).

Item	Decision based on inspection	
	Accept	Reject
Conforming	Correct decision	Wrong decision Type-I error
Non-conforming	Wrong decision Type-II error	Correct decision

3.2.5 Inspection time

In manufacturing processes the inspection time plays a major role in the total manufacturing costs. Longer inspection times constrain the inspection capacity, which may cause increased inspection errors and create new queues in the system that might increase work in-progress

(WIP) and flow. The inspection time for each inspection station can be represented by the inspection cost (Shiau, 2002, Shiau et al., 2007).

3.2.6 Cost components

Producing high-quality products at low cost is always one concern for a multistage manufacturing system. If products are manufactured to an acceptable quality level, then all manufacturing costs will be recovered and the manufacturer will be rewarded with a net profit. For that reason, most of the studies chose to focus on specific cost components related to quality failures (both internal and external failures) and inspection (Raz, 1986 and Mandroli et al., 2006). Internal failure costs are associated with failures and defects of processes, equipment, products, and product materials that fail to meet quality standards or requirements. External failure costs are generated by defective products, and processes during customer use. They include warranties, complaints, replacements or recalls, repairs, poor packaging, handling, and customer returns. Other costs included are inspection cost and manufacturing cost. Inspection cost only occurs when an inspection station is located after the workstation; otherwise the manufacturing cost will take place. The inspection cost is a sum of the fixed cost and the variable cost, and was described in chapter 2.

In summary, characteristics of the AOIS problem were described. All of these characteristics will be considered in the developed GCM, such as costs of inspection, manufacturing, internal and external failure. Also, the GCM takes into account the constraints in terms of limited inspection stations.

3.3 General cost model formulation

The multistage manufacturing process to be considered is such that a product is produced through the processing workstations, and manufactured by specialised production operations to create the final product. The processing workstations are considered to be specialised operations or points and, thus, each processing workstation performs a unique function. The

first processing workstation receives the raw materials and the last involves shipment of the final product. Inspection is possible at all points. The purpose of any inspection is to screen out defective items from non-defective ones, in order to avoid processing items if they are already defective. The aim is to locate inspection stations in such a way that the total cost of a product can be reduced. The total cost is defined as the sum of the costs of production, inspection and failures (during production and after shipment). Since each processing workstation is unique, the quality characteristics of that processing workstation will be unique, and therefore the inspection at that stage will be different from that at another stage. At each inspection point, it is possible to perform either no inspection or multiple inspections of each item or batch of input. Once an item is identified by an inspection station as being defective, it is assigned to a repair facility on the same processing workstation. The item is either inspected again to determine that it is not defective or it is repaired in the event that is defective. In either case it is assumed that this repair facility is perfect.

After the repair, the quantity classified as non-defective from the inspection join the repaired items from the repair facility and this total, which is the same as the input entering the inspection point, is input into the next processing workstation stage. This is done in order to avoid any delivery delay, and to ensure that production can meet the required demand. The cycle of production, inspection and repair, for items classified as defective, then repeats itself through all of the processing workstation stages. At the end of the manufacturing system, the external failure cost representing repair and replacement is incurred for each non-conforming unit that exits the system. It is assumed that the developed model works at quasi static equilibrium situations that in reality may last for a business relevant period in time. Therefore, the developed model uses static variables for calculating the costs. Figure 3.3 illustrates the characteristics of the type of multistage system under consideration consisting of *n* processing workstations as follows:

1. The system is considered to be made up of workstations arranged serially, and parts enter the system in batches.
2. Each processing workstation has a specific probability of producing defective parts.
3. Sampling inspection is performed if an inspection station is located after a workstation in the sequence.
4. Only one final product is considered in the system.
5. The product and the process are discrete.
6. The system has a limited number of inspection stations. Each inspection station can be assigned to perform an inspection operation for one or more workstations.
7. Non-conforming items can either be scrapped or repaired on the same processing workstation. At each inspection station there is a specific probability of selecting non-conforming items for rework.
8. Inspection time for each inspection station can be represented by the inspected cost.
9. Two types of inspection errors are considered in the system: probability that the *k-th* inspection operation classifies a conforming unit (CU) erroneously as a non-conforming unit (NCU) (α_m); and probability that the *k-th* inspection station classifies an NCU as a CU erroneously (β_m).

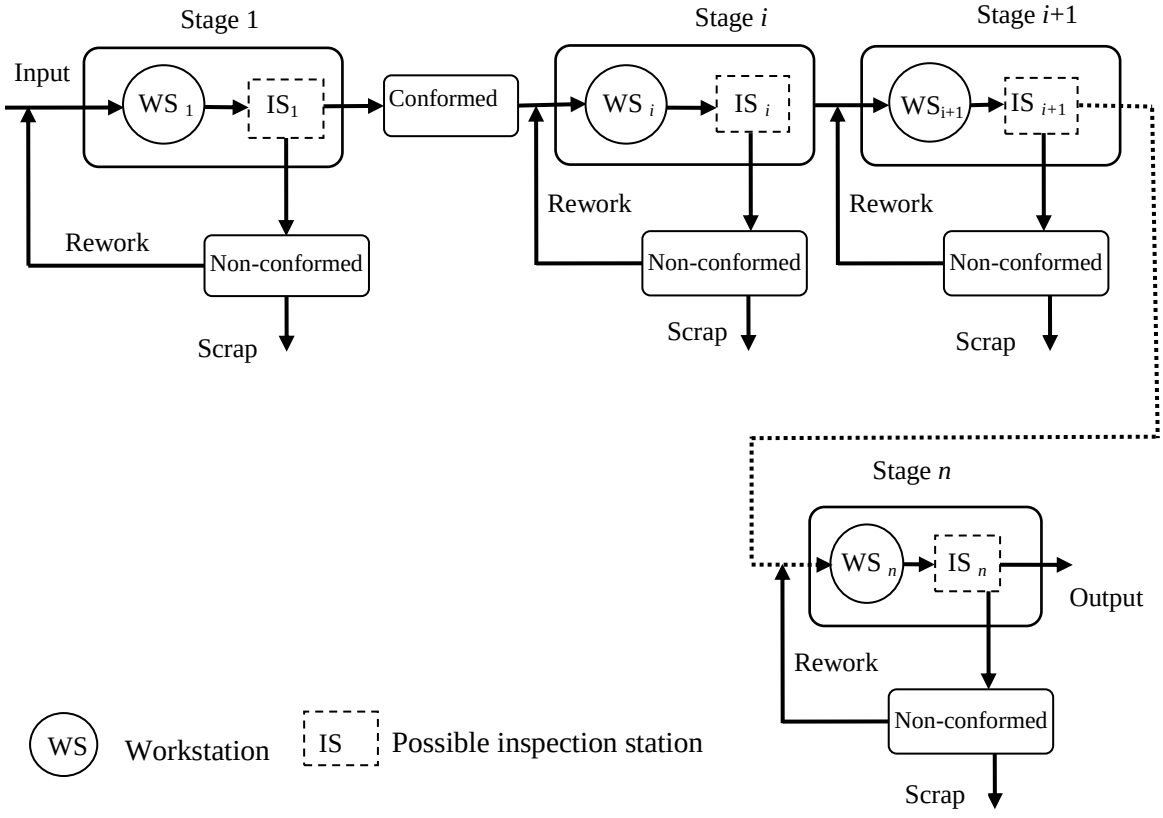


Figure 3.3: A serial multistage manufacturing processes

The significance of the contribution in this chapter is that the general cost model is developed to be very different from those in the literature review. The main differences are:

1. Previous cost models were used full inspection if an inspection station is located after a workstation. The GCM uses sampling inspection plan to perform inspection. This leads to the development of total inspection cost and number of parts for reworking models which are very different from those in the previous literature.
2. Previous cost models were represented external failure cost items as aggregate or only include one of them. The GCM considers all items of external failure cost. This leads to the development of external failure cost model which is very different from those in the previous literature.
3. Previous cost models used the total cost per input unit and the total cost per output unit as the objective function. The GCM is developed such that the number of conforming parts can be computed at each processing workstation and the optimality is defined in terms of minimising cost per item accepted by the customer. This optimality is very different from those cost models in the previous literature.
4. Simplified assumptions which were introduced in those cost models in the literature review have led to the lack of generality. However, the GCM is developed to include all characteristics of the AOIS problem which is very different from those in the previous literature and to maintain the generality.

3.3.1 Inspection stations

For the multistage manufacturing system being considered, the model is developed based on assumption that there is a limited number of inspection stations available (e.g. limited budget) and an inspection policy at each workstation can be described as given by equation (3.1).

$$V_{km} = \begin{cases} 1 & \text{if an inspection station } m \text{ is assigined} \\ & \text{to inspect parts leaving workstation } k \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

3.3.2 Workflow analysis

Consider the number of conforming parts departure a workstation or inspection location, and the number of defective parts entering the following processing workstation or inspection station. These flow constraints can be classified into three classes: good parts; defective parts; and reworked parts. The number of good parts produced at processing workstation *k*, is equal to the number of parts flowing out of the immediately preceding inspection station, or, processing workstation (*k*-1). Equation (3.2) represents the first workstation.

$$NG_1 = B \times (1 - Z_1) \quad (3.2)$$

For all other stations the equation is defined recursively by equation (3.3).

$$NG_k = \sum_m V_{km}[NG_{k-1}(1-\alpha_{k-1}) + ND_{k-1} \times \beta_{k-1} + R_{k-1}](1-Z_k) + (1-\sum_m V_{km}) \times NG_{k-1} \quad (3.3)$$

The number of defective parts produced at first workstation is given by equation (3.4).

$$ND_1 = B \times Z_1 \quad (3.4)$$

For all other stations, is given recursively by equation (3.5).

$$ND_k = \sum_m V_{km}[NG_{k-1} \times \alpha_{k-1} + ND_{k-1}(1-\beta_{k-1}) + R_{k-1}] \times Z_k + (1-\sum_m V_{km}) \times ND_{k-1} \quad (3.5)$$

The number of reworking parts is computed as follows: (i) if V_{km}=1 and the number of bad items (b_k) in the sample at workstation *k* is greater than the acceptance number (a_k) (in this case, the sample size is rejected and a full inspection of the rejected batch is performed consecutively in the same workstation); (ii) if V_{km}=0 or number of bad items in the sample (b_k) is less than or equal $a_{k,}$ in this case the number of reworking parts is zero. The number of parts as non-conforming but repairable is given by equation (3.6).

$$R_k = \begin{cases} \sum_m V_{km}[NG_k \times \alpha_m + ND_k(1-\beta_m)] \times \delta_k & \text{if } V_{km} = 1 \text{ and } b_k > a_k \\ 0 & \text{if } V_{km} = 0 \text{ or } b_k \le a_k \end{cases} \quad (3.6)$$

It should be noted that the abbreviations used in the developed cost model (e.g. α_m, β_m) are commonly used by many studies in the literature review (Lee and Unnikrishnan, 1998, Rau and Chu, 2005, Shiau, 2007).

3.3.3 General cost model analysis

Manufacturing cost: the manufacturing cost of the part in workstation *k* is a product of the manufacturing cost per part (U_k) at workstation *k* multiplied by the number of parts processed at a particular workstation. This cost is assumed to be a sum of the material cost, overhead cost, and setup cost. The number of parts processed at workstation *k* is the sum of the number of parts correctly classified as conformed parts following into process workstation *k* from the previous process workstation and the number of non-conformed parts incorrectly classified as conformed parts following into workstation *k* from the previous process workstation. The number of conformed parts (CP_k) is given by equation (3.7).

$$CP_k = \sum_m V_{km}[NG_{k-1}(1-\alpha_{k-1})] + (1-\sum_m V_{km}) \times (NG_{k-1} + ND_{k-1}) \quad (3.7)$$

The number of non-conformed units erroneously considered as conformed units following into workstation *k* from the previous workstation is ND_{k-1} multiplied by the probability that the inspection station after workstation *k* -1 incorrectly classifies a non-conforming unit as a conforming unit. Hence, the number of non-conformed parts (NCP_k) is given by equation (3.8).

$$NCP_k = \sum_m V_{km}[ND_{k-1} \times \beta_{k-1}] \quad (3.8)$$

Hence, the total manufacturing cost (TMC_k) is defined by equation (3.9).

$$TMC_k = [CP_k + NCP_k] \times U_k \quad (3.9)$$

In the case where no inspection station is allocated $V_{km} = 0$, then the total manufacturing cost is computed by equation (3.10).

$$TMC_k = [NG_{k-1} + ND_{k-1}] \times U_k \qquad (3.10)$$

Inspection cost: inspection cost is a sum of the fixed cost and the variable cost. The fixed cost (FC_m) is a sum of the costs connected with test-equipment installation and setup. The variable cost is computed as follows: (i) if V_{km}=1 and the number of bad items (b_k) in the sample at workstation k is greater than the acceptance number (a_k) (in this case, a full inspection of the rejected batch is performed consecutively in the same stage), the variable cost is the total number of conforming parts and the number of defective parts produced at inspection station multiplied by the unit inspection cost. (ii) In case of V_{km}=1 and number of bad items (b_k) in the sample at workstation k is less than or equal a_k, then the variable cost is computed as the number of items in the sample size multiplied by the unit inspection cost. In case of V_{km}=0, no inspection is performed, the inspection cost is zero. The total inspection cost (TIC_m) is given by equation (3.11).

$$TIC_m = \begin{cases} [FC_m + (NG_k + ND_k)](\sum_m V_{km}) \times IC_m & \text{if } V_{km} = 1 \text{ and } b_k > a_k \\ [FC_m + (s_k \times IC_m)] \times (\sum_m V_{km}) & \text{if } V_{km} = 1 \text{ and } b_k \le a_k \\ 0 & \text{if } V_{km} = 0 \end{cases} \qquad (3.11)$$

Internal failure cost: the internal failure cost is the sum of reworking cost and scrap cost. At each inspection station the non-conforming parts can be scrapped, repaired or incorrectly classified as conformed parts. The number of parts as non-conforming but repairable was given by equation (3.6). The rework cost is given by equation (3.12).

$$RC_k = R_k \times g_k \qquad (3.12)$$

In the model under consideration the allocation of an inspection station means an inspection screen with all the parts being subjected to inspection. Absence of an inspection station is indicated by setting $V_{km} = 0$ .This expression represents the number of non-repairable items produced at workstation *k* on detection subsequent *m* inspection station multiplied by the unit scraping cost at workstation *k*, in case of an inspection station is assigned $V_{km} = 1$, otherwise SC_k =0. The scrap cost is given by is given by equation (3.13).

$$SC_k = \sum_m V_{km}[NG_k \times \alpha_m + ND_k \times (1-\beta_m)] \times u_k \qquad (3.13)$$

and the internal failure cost is given by equation (3.14).

$$IFC_k = SC_k + RC_k \qquad (3.14)$$

External failure cost: this is the cost incurred after the products have been sold to customers. Examples include the cost of replacement and repair. External failure cost (*EFC*), is the sum of the product of the number of defective parts replaced at the customer's end ($W \times ND_k$), the sale price (*P*) of the part and the sum of the product of the number of defective parts repaired at the customer's end (1- *W*) and the direct cost of materials to repair a defective unit (*Y*). Therefore, external failure cost is given by equation (3.15).

$$EFC_k = [W \times ND_k \times P + (1-W) \times ND_k \times Y] \times (1 - \sum_m V_{km}) \qquad (3.15)$$

Total cost: The total cost (*TC*) of processing and inspection of *B* parts in an *n*-stage manufacturing system is given by equation (3.16).

$$\underset{m \in M}{Minimise} \quad TC = \sum_{k=1}^{n} (IFC_k + MC_k + TIC_m + EFC_k) \qquad (3.16)$$

A customer totally sophisticated in quality determination is hypothesised, that is, one who can determine the quality of an item with 100% accuracy. Consequently, in this research, the optimality is defined in terms of minimising cost per item accepted by the customer. The general cost model is developed such that the number of conforming parts can be computed

at each processing workstation. Therefore, the cost for each conforming unit produced is given by equation (3.17):

$$\text{Cost per conforming unit} = TC / NG^{F} \qquad (3.17)$$

where:

$$NG^{F} = NG^{n}(1-\alpha_{n})$$

3.4 Relation between quality and cost

Companies can lose money because they fail to take opportunities to reduce their quality costs. Understanding and improving quality is a key factor in business success, growth, and an enhanced competitive position. Quality costing provides the financial information required, to understand the cost effectiveness of different inspection locations. Rodchua (2006) explains that quality costs have been studied by a number of organisations and it was found that quality commonly costs between 5% and 25% of total sales turnover. The cost of quality adds a significant proportion to the total cost of a part although they add little intrinsic value. Optimisation of the inspection process aims to reduce any unnecessary and avoidable costs, such as the cost of additional manufacturing operations on a defective part. It can also identify quirks and anomalies in cost allocation.

The most generally accepted typology breaks down quality costs into prevention, appraisal, internal failure, and external failure costs. This typology is often referred to as the PAF (prevention, appraisal, and failure). Campanella (1990) defines these costs as follows:

- Prevention costs are ‘the costs of all activities specifically designed to prevent poor quality in products and services’.
- Appraisal costs are ‘the costs associated with measuring, evaluating, or auditing products or services to assure conformance to quality standards and performance requirements’.

- Internal failure costs are ‘the costs resulting from products or services not conforming to requirements or customer/user needs (which) occur prior to delivery or shipment to the customer’.
- External failure costs are ‘the costs resulting from products or services not conforming to requirements or customer/user needs (which) occur after delivery or shipment of the product, and during or after furnishing of a service to the customer’.

In real life, almost all manufacturing processing workstations in multistage manufacturing systems are unable to deliver products with perfect quality. Hence, introducing quality assurance by planning and managing resources, devoted to the inspection and the testing of the critical quality of product features, is a very important issue. Human inspectors, automated inspection, or a combination of both are often used for quality-assurance purposes. These inspection operations may be designed to detect non-conforming items, introduced only at the immediately preceding processing workstation. Otherwise, these inspection stations may be involved in more diagnosis that traces an underlying anomaly that has existed at some previous point, or at all of the preceding processing workstations. The purpose of the inspection is to reduce the total manufacturing cost, resulting from the identification of defective items processed unnecessarily during manufacturing operations. Wild (1989) explains that it would be simply be too expensive to formally inspect items after every process.

Therefore, where to locate inspection stations throughout the process is an important and challenging decision in quality control; that is, striking a balance between minimising the total cost by capturing defective items at the earliest point, and maintaining the required quality of the product.

3.5 Computational time against number of workstations

In AOIS problems, it is clear that a process comprising *n* serial stages offers 2^n possible inspection combinations. In serial multistage manufacturing processes, as the size of the problem grows, so the number of inspection station allocation possibilities increases exponentially. As a result, complete enumeration (CEM) of all combinations becomes prohibitive. Rau et al. (2005) explained that complete enumeration suffers because of the long computation time, especially as the number of workstations increases. Azadeh et al (2012) arrived at the same result that complete enumeration of all combinations becomes prohibitive as the number of stages increases. The CEM approach is based on the premise that all the possible solution combinations are enumerated in the search space, and each one is evaluated individually. The solution combination which has the best objective function value is selected as the solution for implementation. Usually, the complete enumeration method is used as a benchmark for evaluating other optimisation methods. This can only be done when the full enumeration of the search space can be generated in a reasonable time. As shown in Figure 3.4 that the processing time for solving the AOIS problem grows with an increasing number of workstations. The experimental data was approximated using an exponential regression model, and we showed a correlation r = 0.723 and coefficient of determination $r^{2 =}$ 0.967; see equation (3.18):

$$\text{Time} = 0.003 \cdot e^{(0.571 \times WS)} \qquad (3.18)$$

where WS: workstation

This exponential regression equation is the model of the growth of computation time per workstation. The exponential function is used to model the relationship between the number of workstations and the computational time, where a constant change in the independent variable (number of workstations) gives the same proportional change in the dependent

variable (time). Consider a serial multistage manufacturing process producing engine valves. The engine valves are processed in 36 processing workstations using different machines to carry out processes such as cutting, turning, pressing, welding, grinding, heat treatment and plating (Engin, 2008). The firm produces valves that vary in price between $1 and $5, depending on the type of engine.

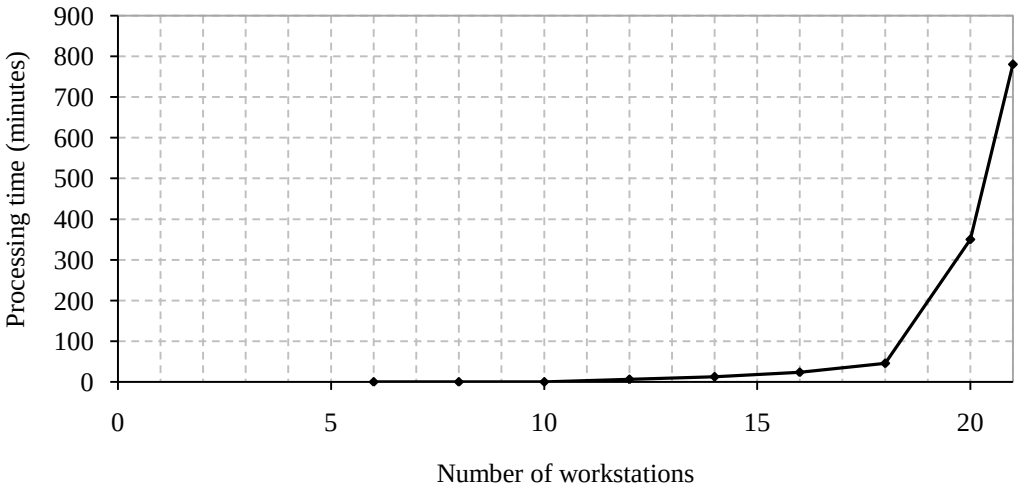


Figure 3.4: Schematic showing the duration of computational time against the number of workstations.

Using equation (3.18), the computation time for the engine valve process consisting of 36 workstations is expected to be 42,299 hours (59 months) on 2.2 GHz CPU and 4GB RAM. With the 88 workstations for the turbine blades described in section 3.1, the time is increased significantly to 3.322×10^{17}hours (4.6139×10^{14} months, about 10^{13} years). This is to be compared to the heuristic algorithm which typically requires an average time about 2 minutes to find a near to optimal solution for the engine valves problem and about 12 minutes for the turbine blade problem on a comparable machine.

Therefore, from an economic aspect it is impractical that the engine valve or turbine blade firms could wait for so long to allocate inspection places by using the CEM. This impracticality of the CEM has led to the use of the heuristic algorithm that sacrifices the

guarantee of finding an optimal solution in order to find a satisfactory solution in a reasonable time. Clearly, the CEM can only be used in problems where the full enumeration of the search space can be generated in a reasonable time. In an industry environment, waste of time means waste of money. In financial terms, using a heuristic algorithm leads to saving money by minimising the total cost of the product, so keeps the company in a good competitive position.

3.6 Computational complexity

Many problems such as the AOIS are combinatorial optimisation problems. Combinatorial optimisation problems involve finding values for discrete variables such that the optimal solution with respect to a given objective function is found. A combinatorial optimisation problem is either a maximisation or a minimisation problem which has associated with it a set of problem instances. The term problem refers to the general question to be answered, which usually has several parameters or variables with unspecified values. The term instance refers to a problem with specified values for all the parameters (Ridge, 2007).

More formally, an instance of a combinatorial optimisation problem Π is a triple (S, f, Ω), where S is the set of candidate solutions, f is the objective function which assigns an objective function value $f(s)$ to each candidate solution $s \in S$, and Ω is a set of constraints (Dorigo and Stützle, 2004). The solutions belonging to the set $\tilde{S} \subseteq S$ of candidate solutions that satisfy the constraints Ω are called feasible solutions. The goal is to find a globally optimal feasible solution s^*. For minimisation problems this consists in finding a solution $s^* \in \tilde{S}$ with minimum cost, that is, a solution such that $f(s^*) \leq f(s)$ for all $s \in \tilde{S}$.

A simple approach to the solution of a combinatorial optimisation problem would be an exhaustive search, that is, the enumeration of all possible solutions and the choice of the best one. However, as described above, in most cases this approach rapidly becomes infeasible

because the number of possible solutions grows exponentially with the instance size n (n=number of workstations). For some combinatorial optimisation problems, understanding the problem structure, and the exploitation of problem-specific characteristics, allows the definition of algorithms that find an optimal solution much more quickly than an exhaustive search does.

When attacking a combinatorial optimisation problem it is useful to know how difficult it is to find an optimal solution. A method of measuring this difficulty is given by the notion of worst-case complexity. Worst-case complexity $O(g(n))$ can be explained as follows: a combinatorial optimisation problem Π is said to have worst-case complexity if the best algorithm known for solving Π finds an optimal solution to any instance of Π having size n in a computation time bounded from above by $const \cdot g(n)$ (Dorigo and Stützle, 2004). In particular, Π is solvable in polynomial time if the maximum amount of computing time necessary to solve any instance of size n of Π is bounded from above by a polynomial in n.

If k is the largest exponent of such a polynomial, then the combinatorial optimisation problem is said to be solvable in $O(n^k)$ time.

An important theory that characterises the difficulty of combinatorial problems is that of NP-completeness. This theory classifies combinatorial problems into two main classes: those that are known to be solvable in polynomial time, and those that are not. Problems in the first class are said to be tractable and problems in the latter intractable. The theory of NP-completeness distinguishes between two classes of problems of particular interest: the class P for which an algorithm exists that outputs in polynomial time the correct answer (‘‘yes’’ or ‘‘no’’), and the class NP for which an algorithm exists that verifies for every instance, independently of the way it was generated, in polynomial time whether the answer ‘‘yes’’ is correct (Dorigo and Stützle, 2004). Clearly, P is a subclass of NP. A problem is said to be NP-hard if every other problem in NP can be transformed to it by a polynomial time

reduction. Intuitively, a polynomial time reduction is a procedure that transforms a problem into another one by a polynomial time algorithm. Therefore, an NP-hard problem is at least as hard as any of the other problems in NP. However, NP-hard problems do not necessarily belong to NP. An NP-hard problem that is in NP is said to be NP-complete. Therefore, the NP-complete problems are the hardest problems in NP: if a polynomial time algorithm can be found for an NP-complete problem, then all problems in the NP-complete class can be solved in polynomial time (Congram, 2000).

3.7 Conclusion

A novel general cost model for solving allocation of inspection stations in serial multistage manufacturing processes was developed. The cost of AOIS problem involves many characteristics, including inspection errors (type I and type II), internal failure cost (rework and scrap), external failure cost (repair and replacement), inspection cost (fixed and variable) and manufacturing cost. The developed general cost model considers all these characteristics together. CEM is guaranteed to find the optimal solution, and prove its optimality for every instance of a combinatorial optimisation problem. In AOIS problem as the size of the problem grows, the number of inspection station allocation possibilities increases exponentially. As a result, CEM of all combinations becomes impractical. This impracticality of CEM has led to the use heuristic algorithm that sacrifices the guarantee of finding optimal solutions for the sake of obtaining good solutions in polynomial-time. In terms of economic aspect, using such heuristic algorithm leads to save money by minimising the computation time. To solve AOIS problem, various metaheuristic methods will be studied and investigated, to select the appropriate method for tackling the AOIS problem. This will be described in the next chapter.

Chapter 4
Optimisation methods characteristics and selection

It is well known that in serial multistage manufacturing processes, as the size of the problem grows, so the number of inspection station allocation possibilities increases exponentially (Valenzuela et al., 2004, Rau and Chu, 2005, Mandroli et al., 2006, Shiau, 2007). To identify the appropriate approach to the AOIS problem, different optimisation methods were investigated and studied. These optimisation methods can be classified as either exact or metaheuristic methods. The optimisation methods will be studied in terms of their characteristics. The aim is to select the appropriate method for tackling the AOIS problem. The following sections give a description of combinatorial optimisation problems and the optimisation methods followed by discussions of their characteristics.

4.1 Combinatorial optimisation problems

Combinatorial optimisation problems involve finding values for discrete variables such that the optimal solution with respect to a given objective function is found. Many optimisation problems of practical and theoretical importance are of combinatorial nature. Combinatorial optimisation problem is either a maximisation problem or a minimisation problem with an associated set of instances. Examples are Travelling Salesperson Problem (TSP), the Quadratic Assignment Problem (QAP) and the Job Shop Scheduling Problem (JSP) (Ridge, 2007). Combinatorial optimisation problems are very hard to solve, which means that very often these problems are classified in the Non-Deterministic Polynomial (NP) class. NP problems are those problems that can only be verified but not be solved in deterministic polynomial time (McCallum, 2005). Within this set there are two related sets, NP-Complete

and NP-Hard. NP-Hard problems are those problems for which all decision problems in NP can be reduced to by a polynomial reduction (McCallum, 2005). If a problem is in the class NP and is NP-Hard then it is called NP Complete.

4.2 Exact methods

The concept of exact methods is based simply on enumerating the full solution space. Some of the most popular exact methods are the Complete Enumeration Method (CEM), Integer Programming (IP), Linear Programming (LP), Non-linear Programming (NLP), Branch and Bound (B&B) and Dynamic Programming (DP). Exact algorithms are guaranteed to find the optimal solution, and prove its optimality for every instance of a combinatorial optimisation problem. However, owing to the inherent complexity of combinatorial optimisation problems, many of them are NP-hard, exact methods that need an exponential run-time in the worst case.

As described in the literature review that complex problems such as the AOIS problem may require exponential time in the worst case. Liang and Smith (2004), Ridge (2007), Volsem and Neirynck (2009) and Azadeh et al (2012) described that using exact methods is infeasible as a result of the exponential size of the solution space with increasing problem complexity. Therefore, one has to resort to using different algorithms and, typically, to sacrifice the guarantee of finding optimal solutions for the sake of obtaining good solutions in polynomial-time. The exact methods are excluded in this chapter and described in Appendix A.

4.3 Metaheuristic methods

A metaheuristic is formally defined as: ‘an iterative generation process, which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions’ (Osman and Laporte, 1996).

Meatheuristics are a more recent attempt to combine basic heuristics into a flexible higher-level framework, in order to better solve complex problems (Ridge, 2007). Some of the most popular metaheuristics for combinatorial optimisation are Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Evolution Strategy (ES), Estimation of Distribution Algorithms (EDAs), Differential Evaluation (DE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and Ant Colony Optimisation (ACO). Many of these metaheuristics have achieved notable successes in solving difficult problems.

4.3.1 Simulated annealing

Simulated annealing is a stochastic search method, emulative of the physical annealing process, where an alloy is cooled gradually so that a minimal energy state is achieved. Kirkpatrick et al. (1983) were the first to propose and demonstrate the application of simulated annealing techniques for combinatorial optimisation problems. SA attempts to solve combinatorial optimisation problems by a process analogous to physical annealing. The analogy associates the set of solutions to the problem with the states of the physical system; the objective function corresponds to the physical energy of the solid, and the ground state corresponds to a globally optimal solution.

While applying SA, a tentative solution s' is generated in each step, which is accepted if the objective function is improved. If the tentative solution s' is worse than the current solution, then it is accepted based on probability, which depends on the objective function difference of the current solution s, new solution s' and parameter *T*, called temperature (Stützle, 1998b). This parameter *T* is lowered (as is also done in the physical annealing process) during the run of the algorithm, reducing the probability of accepting worsening moves. The probability p_{accept} to accept worse solutions is often defined according to the Metropolis distribution, as shown in equation (4.1).

$$p_{accept}(T, s, s') = \begin{cases} 1 & \text{if } f(s') < f(s) \\ e^{(f(s)-f(s'))/T} & \text{otherwise} \end{cases} \quad (4.1)$$

One of the key properties of the algorithm is that, the method sometimes accepts worse solutions than its current best. This gives it a greater probability of getting itself out of local optima (McCallum, 2005). The probability of such a move gradually decreases during the search, until a point is reached whereby only moves that result in high quality solutions are accepted. Because of this, the SA technique has been widely applied for a variety of complex problems. However, El Gamal et al. (1987) and Ram et al. (1996) pointed out that a major disadvantage of the technique, is the need for a great deal of computer time for many runs and also determining the 'cooling schedule' is difficult; for example, what is a sufficient amount of iterations at each temperature? In addition, determining the initial temperature is also difficult. Starting too high will waste computation time, while starting too low will decrease the quality of the search.

4.3.2 Tabu search

Tabu search is an iterative procedure for solving discrete combinatorial optimisation problems. It was first suggested by Glover (1977), has become very popular and been applied to many difficult combinatorial optimisation problems, in a number of different areas. Thiesen (1998) stated that the TS approach has four main elements:

- a local search strategy.
- a mechanism to discourage a return to a recently visited solution, the 'Tabu list'.
- a 'Tabu tenure' policy, used to determine how long a solution will remain in the Tabu list.
- a mechanism to alter the search path when no improvement has been made for some time.

The local search is, usually, a simple, greedy strategy, which finds an improved solution in the immediate neighbourhood of a current solution. Tabu search is equipped with a special

mechanism to avoid being trapped in local optima. It uses a short-term memory to escape from local optima. To prevent the local search returning immediately to a previously visited solution and to avoid cycling, in TS moves to recently visited solutions are forbidden (Glover, 1986). In TS recently visited solutions are added to the Tabu list, and if a solution found by the local search already exists in the Tabu list it is forbidden, and an alternative, non-Tabu, solution will be used. To ensure the search does not quickly exclude all neighbours, solutions are only held in the Tabu list for some period of time. This policy is usually called Tabu tenure. The mechanism used to adjust the search path differs from problem to problem, but a common approach is simply to choose a random solution in the neighbourhood of the current solution. Consequently, Tabu search sometimes allows non-improving solutions to be used for the sake of avoiding local optima.

The Tabu search algorithm, like most of the computational intelligence methods, provides a simple method for solving complex problems. However, the method requires advanced parameter tuning, particularly for Tabu list size (*tl*) and Tabu tenure. Good parameter setting can only be found empirically and requires considerable fine-tuning (Stützle, 1998b).

4.3.3 Genetic algorithm

A genetic algorithm is a metaheuristic strategy inspired by Darwin's principle of natural selection. GA employs random choice to guide a highly exploitative search, striking a balance between the exploration of the feasible domain and the exploitation of good solutions. Genetic algorithms are a specific type of evolutionary algorithms, which are population-based, adaptive search algorithms designed to tackle optimisation problems. They are inspired by models of the natural evolution of species, and use the principle of natural selection for further evolution. Each individual in an evolutionary algorithm represents a solution with an associated fitness value.

The three key functions used in GA are *selection*, *mutation*, and *recombination*. Selection prefers fitter individuals to be chosen for the next generation, and for the application of the mutation and recombination operator. Mutation is a unary operator, aimed at introducing random modifications to an individual. Recombination combines the genetic material of two selected individuals, also called 'parents', by means of a crossover operator to generate new individuals, called 'offspring'. The crossover operator is usually understood as the main operator driving the search in genetic algorithms. The idea of crossover is to exchange useful information between two individuals and, in this way, to generate hopefully better offspring (Langner et al., 2002).

Elitism may be incorporated as a feature where the fittest member of the current generation survives into the next generation. Elitism strategy preserves the best individuals in one generation and translates them to the next generation without any change (Weise, 2009). Kumar and Rockett (2002) described that in a single-objective optimisation, elitism may have the disadvantage of premature convergence, but in the case of multi-objective optimisation the elitism improves the performance. Mutation is understood as a background operator, which introduces small, random modifications to an individual. To keep the population at a constant size the selection operator is used, choosing individuals with preferably higher fitness (survival of the fittest). Finally, the complete cycle of recombination, mutation and selection is called 'generation'.

Genetic algorithms model the natural processes of the inheritance of coded knowledge and survival, by fitness or degree of adaptation to the environment. New application of GA requires only a coding of the problem to this artificial space. However, the quality of such coding is crucial to the genetic algorithms performance. Operating in this space means using problem blind operators that often ignore some important information that could be utilised to guide the search (Janikow, 1993). In addition, determining the size of the population in GA is

a crucial factor. Choosing too small a population size increases the risk of converging prematurely to local minima, since the population does not have enough genetic material to cover the problem space sufficiently. On the other hand, a larger population has a greater chance of finding the global optimum at the expense of more CPU time (Viswanadham et al., 1996).

4.3.4 Evolution strategy

Evolution strategies (ES) were introduced by Rechenberg (1964) and Schwefel (1965). The first version of the algorithm was a so-called (1+1)-ES, in which one parent created one offspring by mutation. The offspring replaced the parent if it had a better fitness. After that, ES has been extended by many studies with a population of μ parents creating λ offspring using both mutation and recombination. The main difference between ES and the generational EAs is the selection procedure. Evolution strategies use deterministic selection whereas selection in generational EAs is probabilistic. Two main selection strategies exist in ES; the $(\mu+\lambda)$-ES and the (μ,λ)-ES. In $(\mu+\lambda)$-ES, the μ parents create λ offspring. The next population is then formed by deterministically selecting the μ best individuals among the available $\mu+\lambda$ individuals. The number of offspring λ is usually less than the total population size, which gives an algorithm with overlapping populations (Dianati et al. 2002). For this reason, $(\mu+\lambda)$-ES is sometimes referred to as a *Steady-state* EA (Ursem, 2003). The other strategy, (μ,λ)-ES, also generates λ individuals from the μ parents. However, in (μ,λ)-ES the parent population is not included in the source population in the selection procedure. The population in (μ,λ)-ES is therefore non-overlapping. Hence, λ must be larger than μ, because individuals are not cloned in ES. Most ES algorithms use self-adaptation to adjust the search process to the problem (Ursem, 2003). The idea in self-adaptation is to encode algorithmic parameters in the genome and use these parameters to

modify the individual. The hypothesis is that good solutions carry good parameters; hence, evolution discovers good parameters while solving the problem.

4.3.5 Estimation of distribution algorithms

Estimation of distribution algorithms (EDAs) are also known as probabilistic model-building genetic algorithms (PMBGA) or iterated density estimation evolutionary algorithms (IDEA) (Shan al., 2006). EDAs explicitly encode the knowledge accumulated in the course of the search into well-structured models, typically probabilistic models, and thus it becomes possible explicitly to exploit that knowledge to improve the efficiency of the search adaptively. More specifically, these models are inductively learnt from good individuals (training examples), and are sampled to create the new individuals of the next generation. A population is not usually maintained between generations, and genetic operators are omitted from EDAs, either partially or completely. Teytaud (2011) stated that the principle of the EDA is to use a probability distribution to represent the potential solutions. Teytaud described how the EDA iteratively estimates the parameters of the distribution by:

- Sampling the domain with the current parameterised distribution.
- Evaluating the sampled points.
- Selecting the best points.
- Rebuilding the probability distribution described by these points.

EDAs are designed to capture the interactions among genes, which represent the internal structure of problem solutions, and in this way they estimate the distribution of good solutions directly, rather than by employing genetic operators.

In practice, EDAs are often combined with other heuristics such as local search, guided local search and genetic algorithms to solve hard optimisation problems (see, for example, Zhang et. al. (2003a, b, 2004)). Hauschild and Pelikan (2011) stated that building explicit

probabilistic models is often more time-consuming than using implicit models defined with simple search operators, such as tournament selection and two-point crossover. That is why it may sometimes be advantageous to use implicit models from conventional evolutionary algorithms instead of explicit models from EDAs.

4.3.6 Differential evaluation

Differential evolution (DE) is one of the recent population-based stochastic evolutionary optimisation techniques. Storn and Price (1995) first proposed DE as a heuristic method for minimising non-linear and non-differentiable continuous space functions. Differential evolution is a population-based search algorithm, which is an improved version of the genetic algorithm. The main reason why better solutions are constructed is that genetic algorithms rely on crossover while DE relies on mutation operation. This main operation is based on the differences of randomly sampled pairs of solutions in the population (Karaboga and Okdem, 2004). In DE, all solutions have the same chance of being selected as parents without reference to their fitness value. The DE algorithm also uses a non-uniform crossover that can take child vector parameters from one parent more often than it does from others. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling there to be a search for a better solution space.

Like other evolutionary algorithms, the first generation is initialised randomly, and further generations evolve through the application of a certain evolutionary operator until a stopping criterion is reached. The optimisation process in DE is carried out with four basic operations, namely initialisation, mutation, crossover and selection (Vanitha and Thanushkodi, 2011).

4.3.7 Covariance matrix adaptation evolution strategy

The covariance matrix adaptation evolution strategy (CMA-ES) adapts a variance covariance mutation matrix used by multivariate normal distributions from which mutations are drawn

(Hansen and Ostermeier, 2001, Hansen et al., 2003). This enables the algorithm to generate correlated mutations, speeding up evolution significantly for many real-world fitness landscapes. Self-adaptation of this mutation matrix is then achieved by integrating information on successful mutations on its recent evolution path, by making similar mutations more likely. Compared to many other evolutionary algorithms, an important property of the CMA-ES is its invariance against linear transformations of the search space. In practice the linear transformation is learned by the CMA algorithm. This is a powerful optimisation procedure and performs especially well in rugged search-landscapes with discontinuities, noise and local optima. The CMA-ES efficiently minimises unimodal objective functions and is, in particular, superior for ill-conditioned and non-separable problems (Auger and Hansen, 2005). Hansen and Kern (2004) showed that increasing the population size improves the performance of the CMA-ES on multimodal functions.

One of the problems with the CMA algorithm, however, is its adhoc and relatively complex nature. Another problem pertains to its sensitivity to local optima (Wierstra et al., 2008). In addition, Omidvar and Li (2011) stated that a disadvantage of the CMA-ES is its relatively high time complexity. This is mainly due to the self-adaptation of the covariance matrix, and Eigen-decomposition. It has been shown that the time complexity of calculating and updating the covariance matrix is of order O (n^3). This makes the CMA-ES more computationally expensive than other EAs.

4.3.8 Particle swarm optimisation

The particle swarm optimisation (PSO) algorithm was first described by Kennedy and Eberhart (1995). It is inspired by the social behaviours of bird flocking and fish schooling. The PSO has recently been applied in many fields because of its simple structure with a small number of parameters, which simplifies the coding of the algorithm. Scientists found that the synchrony of flocking behaviour was achieved through maintaining optimal distances

between individual members and their neighbours. Thus, velocity plays an important role in adjusting for the optimal distance. Furthermore, scientists simulated the scenario in which birds search for food and observed their social behaviour.

Suppose a flock of birds is searching for food in a space where there is only one piece of food available. Each particle's location in the multi-dimensional problem space is a feasible solution to the problem, which is evaluated with a fitness function. A particle in the swarm flies through the space near to its own best flying experience and the flock's flying experience (Azadeh et al., 2012). In other words, the strategy of the bird for finding the food is to change its velocity to fly near the best place that it has already experienced. PSO actually uses both aspects of cooperation and competition among the individuals in the population, which means that it combines a local and a global search to reach the global optima. The distance of the particles to the food is measured by the pre-determined fitness function in all iterations. The particles in a local neighbourhood share their information about their ''best'' positions, and then use the information to adapt their own velocities, and thus update their positions.

In fact, each particle in this swarm has two kinds of intelligence, namely self-intelligence and social-intelligence (Azadeh et al., 2012). The PSO algorithm is similar to other evolutionary algorithms. In PSO, the population is the number of particles in a problem space. Each particle will have a fitness value, which will be evaluated by a fitness function to be optimised in each generation. Each particle knows its best position and the best position so far among the entire group of particles. The local best (*lbest*) of a particle is the best result (fitness value) so far reached by the particle, whereas the global best (*gbest*) is the best result in terms of fitness over the entire population. The particle will have velocity, which directs the flying of the particle. In each generation the position and the velocity of the particles will be updated as given by equations (4.2) and (4.3):

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{4.2}$$

$$v_i^{k+1} = \omega_k \, v_i^k + c_1 \, rand_1 \times (lbest_i - x_i^k) + c_2 \, rand_2 \times (gbest_i - x_i^k) \tag{4.3}$$

The PSO has gained increasing popularity among researchers and practitioners as a robust and efficient technique for solving difficult population-based stochastic optimisation problems. It has been applied successfully to flow-shop scheduling problems (Liao et al., 2007), multiple-level warehouse layout design problems (Önüt et al., 2008) and optimum controller design for automatic voltage regulator (AVR) power systems (Aghababa et al., 2010). PSO is a modern evolutionary algorithm comparable with the genetic algorithm (GA). It is similar to the GA in some aspects, since it starts with a randomly generated population (solution), has a fitness function to evaluate the solutions, and uses random techniques to update the population in all iterations. However in the PSO, unlike the GA, updating the particles depends on their memory and so does not have special operators. It is also important to note that: ‘‘it has a more global searching ability at the beginning of the run and a local search near the end of the run. Therefore, while solving problems with more local optima, there are more possibilities for PSO to explore local optima at the end of the run’’ (Önüt et al., 2008).

4.3.9 Ant colony optimisation

The ant colony optimisation (ACO) algorithm was first proposed in 1991 by Marco Dorigo, in his PhD thesis, ‘Optimisation, Learning, and Natural Algorithms’, and, since then has become very popular after its further publication, for solving the travelling salesman problem (TSP) (Dorigo et al., 1996). ACO algorithms are inspired by the foraging behaviour of real-life ant colonies, in which individual ants drop a chemical substance called a pheromone on the path, while moving from the nest to the food source and back. They communicate information about the food source using pheromones along the ground. In the same way, artificial ants use a memory mechanism to store some numeric information about the states

they visit, achieving the same indirect communication. When an ant finds a food source it returns to the nest. Since ants on a short path will return to the nest more quickly, more pheromones will be dropped on the shorter paths. As a result, paths that are more regularly travelled become more attractive and, by means of that self-strengthening behaviour, will be used more frequently. Owing to this interesting ant behaviour, it was observed that, after some time, a colony of ants would select the shortest path from the nest to the food source (Dorigo et al., 1996). The ant colony optimisation algorithm is based on ant foraging behaviour, as explained above.

Ant Colony Optimisation is a relatively recently developed technique for solving NP- hard optimisation problems (Dorigo and Stützle, 2004). ACO has the advantage of using an adaptive memory, which involves keeping track of recent decisions made and solutions found, or generating synthetic parameters to describe the search. More specifically, it is based on the foraging activity of ants, and can be viewed as both a successful application of swarm intelligence and a metaheuristic.

4.3.10 Conclusion

In summary, owing to the practical importance of optimisation problems, many algorithms have been devised for their solution. These are classified as either exact or metaheuristic algorithms. In general, one is interested in solving the AOIS problem as efficiently as possible, where ‘efficient’ usually means as fast as possible. In fact, as a result of the high complexity and difficulty of optimisation problems, often exact approaches (that guarantee to find the optimal solution) are only feasible for small size problems, and can require a lot of computational effort. In contrast, different approaches based on metaheuristics are described, which are capable of finding, good and sometimes optimal, solutions to complex problems, in a generally shorter computation time (Bianchi, 2006). In addition, metaheuristics are designed to be general purpose algorithms that can be applied without major modifications to

many problems. The characteristics for exact and metaheuristic optimisation methods will be discussed in the next sections. The aim is to find the suitable approach to solve the AOIS problem.

4.4 Classification of metaheuristics

Metaheuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. Many of the metaheuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in metaheuristic algorithms randomness is not used blindly but in an intelligent and biased form (Birattari et al., 2001). There are different ways to classify and describe metaheuristic algorithms. The most common way of classifying metaheuristics used by Birattari et al. (2001), Blum and Roli (2003), Mills et al. (2003), Ridge (2007) and (Rajab , 2012) are summarised as follows:

- **Memory usage versus memory-less methods**. One of the possible characteristic of metaheuristics is the use of the adaptive memory to influence the future search direction. Memory is explicitly used in tabu search. Short term memory is used to forbid revisiting recently found solutions and to avoid cycling, while long term memory is used for diversification and intensification features. Metaheuristics without adaptive memory determine their next action solely on the current state of their search process. This means that they do not have the ability to memorise traces that they used a few cycles before (Ridge, 2007, Mills et al 2003). In ant colony optimisation an indirect kind of adaptive memory of previously visited solutions is kept via the pheromone trail matrix which is used to influence the construction of new solutions. Also, the population of the genetic algorithm could be interpreted as a kind of memory of the recent search experience. On the contrary, simulated annealing does not use memory functions to influence the future search direction and therefore is memory-less algorithm.

- **Population-based versus single-point search**. Some (metaheuristic) methods use a single-point search to create a solution, while other methods use a population. If the algorithm is working in a population solution, it is a population based algorithm. Otherwise, it is single point search algorithm, where sometimes it is called a trajectory algorithm like Tabu search (Blum and Roli 2003, Rajab, 2012).The advantage of using a population is to increase the exploration of the search space. However, the performance of the method depends strongly on the procedure of the population manipulated. Population-based methods evolve a set of points in the search space. In the single-point search, only one single solution is manipulated at each iteration of the algorithm. Tabu search and simulated annealing are such single-point search methods. On the contrary, in ACO algorithms, PSO, and GA, a population of ants, particles or individuals, respectively are used.
- **Dynamic versus static objective function**. Metaheuristics can also be classified according to the way they make use of the objective function. Dynamic metaheuristics modify the fitness landscape, as defined by the objective function, during search to escape from local minima (Birattari et al., 2001). Tabu search may be interpreted as using a dynamic objective function, as some points in the search space are forbidden, corresponding to infinitely high objective function values. Yet, all the other algorithms introduced so far use a static objective function.
- **One versus various neighbourhood structures**. Most local search algorithms are based on one single neighbourhood structure which defines the type of allowed moves. In other words, the fitness landscape topology does not change in the course of the algorithm. This is the case for simulated annealing and tabu search. Some metaheuristics allow swapping between different fitness landscapes to help diversify search. Others operate on one neighbourhood only. The mutation operator in genetic algorithms may also be interpreted as a change in the neighbourhood during the local search. Applications of the crossover operator

have been interpreted as moves in hyper-neighbourhoods (Birattari et al., 200, Rajab, 2012), in which a cluster of solutions in genetic algorithms these clusters are of size two is used to generate new solutions. On the other side, the solution construction process in ant colony optimisation is not based on a specific neighbourhood structure. Nevertheless, one could interpret the construction process used in ACO as a kind of local search, but this interpretation does not reflect the basic algorithmic idea of these approaches.

- **Nature-inspired versus non nature-inspired**. A minor point for the classification of metaheuristics is to take into account their original source of inspiration. The algorithmic approaches try to take advantage of these phenomena for the efficient solution of combinatorial optimisation problems (Ridge, 2007). There are nature-inspired algorithms, like genetic algorithms, simulated annealing, particle swarm and ant algorithms, and non nature inspired ones such as tabu search. This dimension is of little use as most modern metaheuristics are hybrids that fit in both classes.

Table 4.1 presents the characteristics of the studied optimisation methods. The following notations are used: (●) means that the characteristic is present, (◒) that this characteristic is partially present, and (○) that the characteristic does not appear. It should be noted that may not all implementations of these algorithms correspond to this classification, but it rather gives an indication of the particular characteristics of these methods in their "standard" use.

Metaheuristic methods guarantee of finding optimal solutions is sacrificed for the sake of getting good solutions in a significantly reduced amount of time. Providing a balance between the exploitation and the exploration of a given optimisation problem is the most important characteristic for any metaheuristic technique. The exploitation is the accumulated search experience. The exploration is to identify regions of the search space, with high quality solutions in a problem. The core difference between the metaheuristics concerns the particular way in which they try to reach this balance (Birattari et al., 2001). For example,

PSO algorithms combine the local search methods (via self experience) with the global search method (via neighbouring experience), attempting to balance exploration and exploitation (Cruz et al., 2004). In ACO the searching behaviour of ant algorithms can be characterised by two main features, exploration and exploitation. Exploration is the ability of the algorithm to broadly search through the solution space, whereas exploitation is the ability of the algorithm to search thoroughly in the local neighbourhood, where good solutions have previously been found. Higher exploitation is reflected in rapid convergence of the algorithm to a suboptimal solution, whereas higher exploration results in better solutions at higher computational cost due to the slow convergence of the method (Moeini and Afshar, 2009). Every metaheuristic approach should be designed with the aim of effectively and efficiently exploring a search space. The search performed by a metaheuristic approach should be intelligent enough to both intensively explore areas of the search space with high quality solutions, and to move to unexplored areas of the search space when necessary.

The concepts for reaching these goals are nowadays called intensification and diversification (Blum and Roli, 2003). The main difference between intensification and diversification is that during an intensification stage the search focuses on examining neighbours of elite solutions. The diversification stage on the other hand encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. In ACO algorithms, diversification is achieved through the application of pheromone re-initialisation. Intensification is achieved by letting the restart-best solution or some ant of the elitist list deposit pheromone. In TS algorithm intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good. They may also initiate a return to attractive regions to search them more thoroughly.

Table 4.1: Characteristics of meatheuristic methods

Methods	Characteristics				
	Memory usage	Population	Dynamic $f(x)$	Multiple neighbourhoods	Nature-inspired
Simulated annealing	○	○	○	○	●
Tabu search	●	○	◒	○	○
Genetic algorithm	◒	●	○	◒	●
Particle swarm	●	●	○	○	●
Evolution strategy	◒	●	○	◒	●
Evolution of distribution algorithms	◒	●	○	◒	●
Differential evaluation	◒	●	○	◒	●
Covariance matrix adaption evolution strategy	◒	●	○	◒	●
Ant colony	●	●	○	○	●

Since elite solutions must be recorded in order to examine their immediate neighbourhoods, explicit memory is closely related to the implementation of intensification strategies.

It can be seen from Table 4.1, that some of these (metaheuristic) methods have additional characteristics when compared to the other techniques. These characteristics are memory usage and population, and may lead to enhancing the performance of the method. As a result, a high quality solution can then be obtained. For example, Tabu search uses a short-term memory to escape from local minima, whereas in ant colony methods, the ants keep in their memory the partial solution built by leaving pheromones on the path they have traversed on the construction graph. From one generation to the next, a global memory is updated that guides the construction of solutions in the successive population. The best solutions found so far by the ants are used for the memory update. In evolutionary methods, the population of a genetic algorithm could be interpreted as a kind of memory of the recent search experience. This characteristic (memory) will guide these methods to identify regions of the search space with high quality solutions. In PSO, the particles of the swarm fly through hyperspace and have two essential reasoning capabilities: their memory of their own best position *local best (lbest)* and knowledge of the global or their neighbourhood's best *global best (gbest)*. In CMA-ES, selection and recombination are the leading operations.

Some meatheuristic methods are considered as trajectory methods and others are defined as discontinuous walk methods. The difference between them is the use of a population of search points, or the use of one single search point. In the latter, only one single solution is manipulated at each iteration of the algorithm. For example, Tabu search and simulated annealing are single-point search methods; in every cycle, one single solution is created. On the other hand, a population of ants is used for ant colony algorithms, particles are used in PSO and individuals are used in genetic algorithms. However, the performance of a particular method depends strongly on the way the population is manipulated (Birattari et al. 2001).

In terms of representation of the problem ant algorithms have the advantage of being easier to set to problems where there exists an explicit graph representation (McCallum, 2005). In ACO algorithms artificial ants are a stochastic constructive procedure that incrementally builds a solution by adding opportunely defined solution components to a partial solution under construction. Therefore, the ACO metaheuristic can be applied to any combinatorial optimisation problem for which a constructive heuristic can be defined (Dorigo and Stützle, 2004). In addition, Selvi and Umarani (2010) stated that the main advantage of the ACO is the Positive Feedback accounts for rapid discovery of good solutions. Also, it can be used in dynamic applications. On the other hand, the same authors stated that the PSO method easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction.

4.5 Conclusion

It can be seen from Table 4.1, that the characteristics of the optimisation methods were classified and described. Ant colony optimisation and particle swarm are the two methods which have the most desirable characteristics of all the methods. To select one of the two methods it should be tested that whether it is suitable or not for the AOIS problem. Therefore, it is important to understand the difficulty of the problem using fitness landscape. It should be noted that to date none of the surveyed studies in chapter 2 investigated fitness landscape for the AOIS problem. Understanding the geometry of the landscape for the AOIS problem helps for selecting the appropriate algorithm. The experimental results in section 5.5.2 show that the ACO is well suited for the AOIS problem than the PSO algorithm. In ACO, the search process can be used to identify promising regions of the search space, with high quality solutions. This can be done by using the pheromone trails as an adaptive memory of solution components, which have been part of the best local minima found so far. The ACO approach was rather unexplored for the AOIS problem, at the time this research started. The ACO

algorithm will be joined to a local search method to improve the performance of the algorithm. The biological inspiration of ant colony optimisation and how it is transferred into the algorithms will be described in detail in the next chapter.

Chapter 5
Ant colony optimisation

This chapter introduces the biological inspiration of ant colony optimisation along with how it is transferred into algorithms. This chapter also explains how biological ants find short paths under controlled experimental conditions, and illustrates how the observation of this behaviour has been translated into working optimisation algorithms. The general construction of the ant system algorithm and its extensions also will be explained. The application of ant colony optimisation (ACO) to solve a variety of combinatorial optimisation problems is presented. The importance of heuristic information and the local search method for ACO is described in this chapter.

5.1 The biological inspiration

The ant colony optimisation algorithm was first proposed in 1991 by Marco Dorigo with his PhD thesis "Optimisation, Learning, and Natural Algorithms" and has since become very popular after its publication for solving the well-known travelling sales-man problem (TSP) (Dorigo et al., 1996). Many researchers have since developed ACO variants for tackling well-known NP-hard problems, and have applied them to a range of different problems such as telecommunication networks (Di Caro and Dorigo, 1998), quadratic assignment problems (QAP) (Stützle and Dorigo, 1999), scheduling problems (Kumar et al., 2003), vehicle routing problems (VRP) (Bell and McMullen, 2004), flexible manufacturing scheduling (Rossi and Dini, 2007), graph colouring (Bui et al., 2008), assembly line balancing (Baykasoğlu and Dereli, 2009) and layout planning (Ning et al., 2010). A more exhaustive list of the variety of applications can be found in Blum (2005).

Ants can smell pheromones and they tend to choose, stochastically, paths marked by strong pheromone concentrations. Ants release these chemical substances on the path while moving from the nest to food sources and back. The pheromone trail-laying and following behaviour of some ant species to find the shortest paths has been investigated in controlled experiments by several researchers. These experiments employed in what have come to be known as the single and double bridge experiments. The single bridge experiment was designed and run by Deneubourg et al. (1990), which was cited as the origin of Dorigo's work. Goss et al. (1989) used a double bridge experiment connecting a nest of ants of the Argentine ant species (*Iridomyrmex humilis*) and a food source.

Figure 5.1(a) shows the experimental setup for the single bridge experiment. In the experiment, the nest of a colony of ants and a food source are separated by a binary bridge in which each branch has the same length. Ants are then free to move between the nest and the food source. The percentage of ants which choose one or the other of the two branches is observed over time. In the experiment, there is initially no pheromone on the two branches. Therefore, the ants do not have a preference and they select the branches with the same probability. Due to random fluctuations, a few more ants will select one branch over the other. As ants release pheromones while walking, a greater number of ants on one branch results in a larger amount of pheromone on that branch. This larger amount of pheromone in turn encourages more ants to select that branch. Finally, the ants converge on one single path, as shown in Figure 5.1(b) (Dorigo and Stützle, 2004).

The experiment above can be modified so that the branches of the bridge are of different lengths (Goss et al., 1989). In this case, the first ants to arrive at the food source are those that take the shortest branch. Consequently, when they start on the return leg, more pheromone is present on the short branch than on the long branch, stimulating successive ants to choose the short branch. In this case, the importance of initial random fluctuations is much reduced, and

the stochastic pheromone trail following the behaviour of the ants coupled to differential branch length is the main mechanism at work.

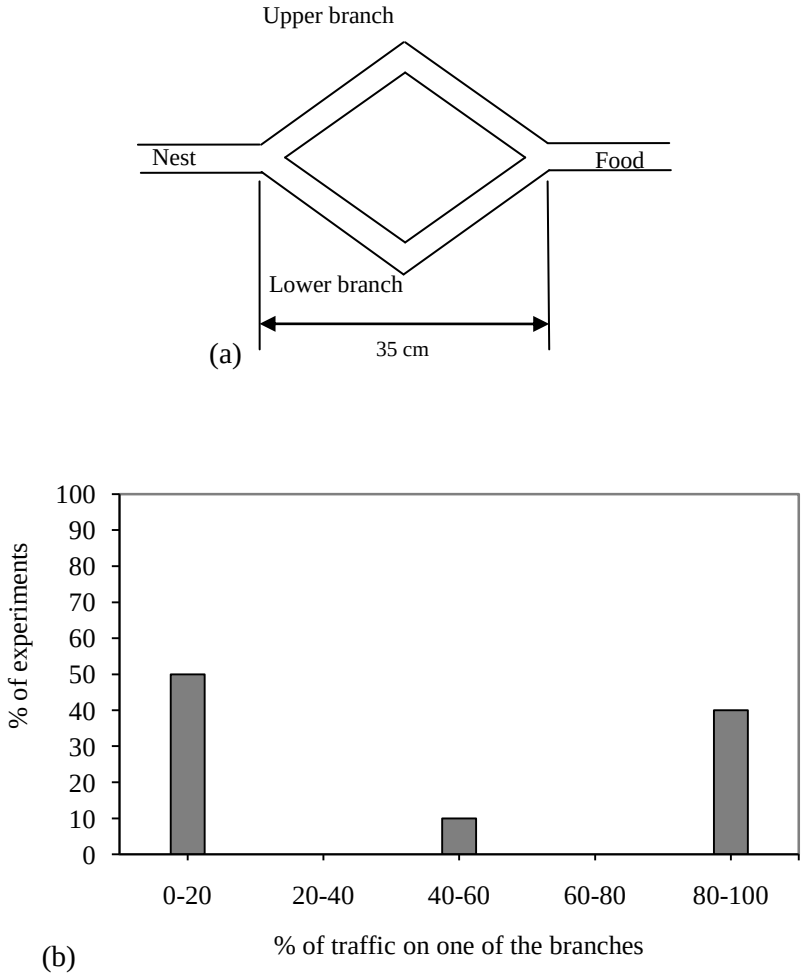


Figure 5.1: Single bridge experiment (Deneubourg et al., 1990). In this case, the ants use one branch or the other in approximately the same number of trials.

Figure 5.2 shows the experimental setup and the results of an experiment with a double bridge having branches of different lengths. In Figure 5.2 (a), the ants are shown as they start exploring the double bridge. Figure 5.2 (b) shows the distribution of the ants further into the experiment. Here, most of the ants have converged on the shortest path. Figure 5.2 (c) shows the distribution of the percentage of ants that selected the shorter path.

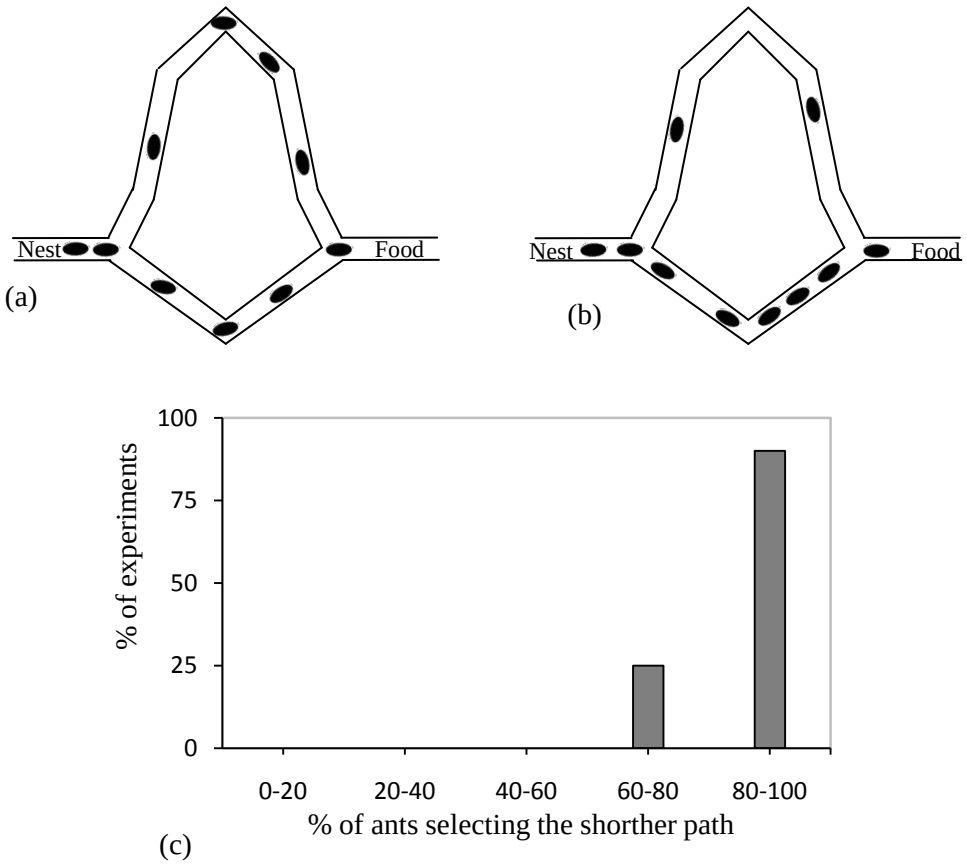


Figure 5.2: Double bridge experiment (Goss et al., 1989); in all trials, the vast majority of ants chose the short branch.

When the two experiments are compared, the influence of initial random fluctuations in the second experiment is much reduced. The results of the two experiments were used to construct a probabilistic model of ant foraging behaviour which forms the core of the ant colony optimisation metaheuristic. Interestingly, it was found that even when the long path is twice as long as the short one, not all ants use the short path, but a small proportion may take the longer one. This may be interpreted as a type of ‘‘path exploration’’.

5.2 Ant colony optimisation metaheuristic

Ant colony optimisation is a metaheuristic in which a colony of artificial ants cooperates in finding good solutions to difficult discrete optimisation problems. The term *metaheuristic* was first coined by Glover in 1986 in the first publication of Tabu Search. The term refers to

‘‘a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality’’ (Glover and Laguna, 1997). Cooperation is a key design component of ACO algorithms. The choice is to allocate the computational resources to a set of relatively simple agents (artificial ants) that communicate indirectly by stigmergy, that is, by indirect communication mediated by the environment. ACO algorithms can be used to solve combinatorial optimisation problems.

5.2.1 Problem representation

Ant colony optimisation algorithms are based on a parameterised probabilistic model that is used to model the pheromone trails. Artificial ants build their solutions incrementally by performing random walks on a connected graph G=(C, L) where C is the set of nodes and L is the set of connecting arcs, as shown in Figure 5.3. Each routing through the graph G defines a unique vector of solution components. When a constrained version of the problem is considered, the problem constraints are built into the ant’s constructive procedure in such a way that in every step of the construction process only feasible solution components can be added to the current partial solution. The ants use pheromone values attached to the arcs connecting the nodes of the construction graph to make stochastic decisions on how to traverse it. The value of this probability parameter for the arc connecting any two nodes *i* and *j* is denoted by τ_{ij}.

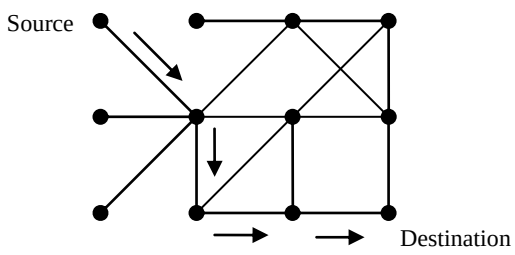


Figure 5.3: Ants construct solutions by building a path from
a source to a destination node (Dorigo and Stützle, 2004)

5.2.2 The metaheuristic

Many ACO algorithms follow a standard form and can be viewed as three procedures, generally referred to as *ConstructAntsSolutions*, *UpdatePheromones* and *DaemonActions* (Dorigo and Stützle, 2004). Depending on the application and the designer of an algorithm, these procedures can be scheduled and synchronised in any number of ways. The ACO framework is outlined in the following procedure:

Procedure *Ant Colony Optimisation*
Initialise pheromone trails, calculate heuristic information
WHILE termination conditions not met **DO**
ConstructAntsSolutions
UpdatePheromones
DaemonActions {optional}
ENDWHILE
End *Ant Colony Optimisation*

ConstructAntsSolutions: An ant builds a solution incrementally by moving through the nodes of the graph *G* as shown in Figure 5.3. Ants travel by applying a stochastic local decision strategy that makes use of pheromone trails (τ_{ij}) and heuristic information (η_{ij}) on connections of the graph. While moving, the ant keeps in memory the partial solution it has built in terms of the path it has traversed on the graph. Previously visited nodes are considered off-limits in future construction steps. Subsequently, starting from the current node, an ant moves to still unvisited nodes according to the probability distribution as shown in equation (5.1) until a tour is completed.

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{r \in N_i^k} [\tau_{ir}(t)]^{\alpha} \times [\eta_{ir}]^{\beta}} & \text{if } j \in N_i^k \\ 0 & \text{if } j \notin N_i^k \end{cases} \qquad (5.1)$$

Where N_i^k is the neighbourhood of ant *k* when in node *i*. The pheromone trails and heuristic information are weighted via two parameters, α and β. They determine the relationship between the heuristic information and the pheromone and are always positive. The neighbourhood of node *i* contains all the nodes directly connected to node *i* in the graph G=(C, L), except for the predecessor of node *i*. In this way, the ants avoid returning to the same node they visited immediately before node *i*. An ant repeatedly hops from node to node using this decision policy until it eventually reaches the destination node. Due to differences among the ants' paths, ants travelling on shorter paths will reach their destinations faster.

In ACO algorithms, artificial ants are stochastic solution construction procedures, which are biased by artificial pheromones and heuristic information. In fact, in the initial stages of the search, the pheromones, being set to initial random values, do not guide the artificial ants in a useful way. This leads to creating tours of very poor solution quality. The main role of heuristic information is to avoid this, by initially biasing ants so that they can build reasonably good tours from the very beginning search of the algorithm. Heuristic information can be derived from a problem instance to guide ants in the solution construction process. The heuristic information is defined in accordance with the characteristics of the problem that is yet to be solved, on a case-by-case basis. In many cases, the heuristic information is the cost, or an estimate of the cost, of adding the component or connection to the solution under construction (Dorigo and Stützle, 2004).

- **Heuristic information review**

Several types of heuristic information are applied in various algorithms. For example, in the TSP, the distance between cities is an obvious and computationally inexpensive heuristic to use; in other problems, it may be much more difficult to find, or too expensive to compute,

meaningful heuristic information which helps to improve performance (Dorigo and Stützle, 2004).

Stützle and Hoos (2000) described that the ant system and all other ACO algorithms for the TSP use heuristic information $\eta_{ij} = 1/d_{ij}$; that is, the heuristic desirability of going from city *i* directly to city *j* is inversely proportional to the distance between the two cities.

Liang and Smith (2004) used an ACO method to solve the redundancy allocation problem (RAP). The RAP is a series system of *s* independent-out-of-*n* subsystems as shown in Figure 5.4. A subsystem *i* is functioning properly if at least k_i of its n_i components are operational. A series-parallel system is where k_i=1 for all subsystems. In the formulation of a series-parallel system problem, for each subsystem, multiple component choices are used in parallel.

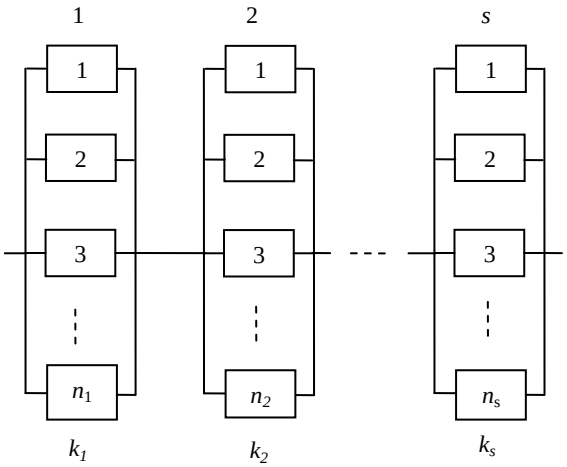


Figure 5.4: Series-parallel system configuration (Liang and Smith, 2004)

The aim is to select the optimal combination of the components and redundancy levels to meet system level constraints while maximising system reliability. Heuristic information is used as follows:

$$\eta_{ij} = \frac{r_{ij}}{c_{ij} + w_{ij}}$$

where r_{ij}, c_{ij} and w_{ij} represent the associated reliability, cost and weight of the component for the subsystem. Components with higher reliability, and lower cost and weight have a greater probability of selection.

Spiliopoulos and Sofianopoulou (2008) applied ACO for the manufacturing cell design problem. The aim was to decentralise and create manufacturing cells by grouping the machines into clusters and the various parts into part families. After that, the processing of each part family was allocated to a single machine cluster. As a result, the processing times, transport and queuing can be reduced and the need for frequent set-up can be eliminated. They proposed heuristic information which is used to calculate the attractiveness of assigning machine *i* to every candidate cell *k* that is not already occupied in full as follows:

$$\eta_{ik} = \frac{1}{1 + \sum_{\substack{j=1 \\ j \notin k}}^{n} c_{ij}} \qquad k = 1,...K \quad k \notin tabu$$

The *tabu* set contains the cells that are saturated. Also, $j \notin k$ denotes that machine *j* is not allocated to cell *k*. Therefore, the sum in the denominator is the "external" total cost between machine *i* and machines already allocated to cells other than *k*. The smaller this value, the more attractive is the allocation of machine *i* to cell *k*.

Ning et al. (2010) applied the max-min ant system to solve the construction site layout planning (CSLP) problem. The max-min ant system will be described in more detail in section 5.3.4. CSLP is a dynamic multi-objective optimisation problem as there are different facilities employed in the different phases of a construction project. The CSLP problem was modelled as a QAP under two objective functions of minimising the representative score of safety/environment concerns and the total handling cost of interaction flows between the facilities associated with the construction site layout. They used heuristic information to assign facility *i* to location *j* as follows:

$$\eta_{ij} = \frac{1}{f_i \times d_j}$$

where the two vectors f_i and d_j represent the flow potential of facility i and the distance potential of location j, respectively. They are calculated by the sum of the flows (closeness relationship) from facility i to all other facilities and the sum of the distances from location j to all other locations, respectively. The lower the value of d_j, the more central the location is considered to be, and the higher the value of f_i, the more important the facility is considered to be.

It is concluded that heuristic information is important for ACO and helps to improve the performance of ACO. It can be seen from the preceding review that each type of problem should have a specific type of heuristic information. According to the literature review, this heuristic information has not yet been constructed for the allocation of inspection station (AOIS) problem, because this is the first time that the ACO method has been applied to the AOIS problem. There is a need to create appropriate heuristic information for the AOIS problem. In this research, a novel heuristic information method for ACO will be constructed for the AOIS problem and presented in chapter 6.

UpdatePheromones: Pheromone values are subject to update and change dynamically in the course of programme execution, while heuristic values usually remain static throughout the search. The set of pheromone values represents the memory of the algorithm; the set of heuristic values indicates the desirability of going from node *i* to node *j*. This kind of pheromone update is called an online step-by-step pheromone update. Once an ant has built a solution, it can (by using its memory) retrace the same path backward and update the pheromone trails of the used connections according to the quality of the solution it has built. This is called an online delayed pheromone update. The solution quality is gauged by evaluating the objective function for the given solution components (input variable values).

The better the solution quality, the higher the amount of pheromone will be deposited on the arcs connecting the nodes visited in the construction of the solution.

A further essential concept in ant colony optimisation is pheromone evaporation. Pheromone evaporation is the process through which the strength of the pheromone trail on the components decreases over time. Pheromone evaporation is very important to avoid a very quick convergence of the algorithm toward the near-optimal region. It implements a useful form of forgetting, favouring the exploration of new areas in the search space.

DaemonActions: Daemon actions can be used to implement centralised actions which cannot be performed by a single ant. In other words, it represents the situation where some extra commands need to be processed on a global scale. A good example of a daemon action is adding a local search to the ACO algorithm (Dorigo et al., 1996). The importance of adding a local search to the ACO algorithm will be described in section 5.4.

This framework is implemented by many ant colony algorithms and it is important to view each algorithm with its motivating domain to gain an understanding of the differences that define them as separate algorithms.

5.3 Ant system (AS)

This algorithm explains a number of characteristics, positive feedback, a distributed architecture and a solution construction procedure. It is based on three initial attempts at defining the algorithm: Ant-Density, Ant-Quantity (Dorigo et al., 1991a) and Ant-Cycle (Colorni et al., 1992). The three original algorithms were applied to the TSP and differed in the amount of pheromone laid and the timing of the trail update. Ant-Density used a constant update after every step an ant took. Ant-Quantity used an amount proportional to the distance between cities i and j ($\frac{Q}{c_{ij}}$, where Q was an arbitrary parameter and c_{ij} was the cost of

moving from city *i* to city *j*). Ant-Cycle was the first to perform the trail update at the end of the construction process and was updated with a value proportional to the length of the tour, $\frac{Q}{L_k}$, where L_k was the length of the *k*-th tour. Ant system implements the Ant-Cycle method of updating at the end of the ant generation and activity function in the Procedure ACO Metaheuristic. The main components of the algorithm are the ants' solution construction and the pheromone update. An ant chooses to move from the current node to the next adjacent node based on a rational combination of two factors, namely the heuristic information (η_{ij}) of that move and the quantity of pheromone on the edge (τ_{ij}) which is to be traversed. The pheromone matrix (τ) is the memory of the algorithm, allowing indirect communication between ants. The next node is selected based on the probability transition rule as described in equation (5.1). Equation (5.1) is an equation defining how new solutions are integrated into the pheromone matrix. After all the ants have constructed their tours, the pheromone trails are updated using equations (5.2) and (5.3).

$$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad (5.2)$$

where *m* is the number of ants and ρ is a parameter that reduces the pheromone on unused edges, sometimes referred to as the learning rate or pheromone decay.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f(s_k)} & \text{if arc}(i, j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases} \qquad (5.3)$$

where *Q* is the total quantity of pheromone per unit length of the trail laid on the edge (*i*, *j*) by the *k*-th tour at time *t* and $f(s_k)$ is the length of the *k*-th ant's tour.

Initial experiments conducted on a set of benchmark problems applied by Dorigo et al. (1991b) found that the performance of *Ant-Cycle* was much better than the other two

algorithms. As a result, research on AS focused on improving the characteristics of *Ant-Cycle*, which is now known as the Ant System, while the other two algorithms were ignored.

The Ant System was applied to the QAP by Maniezzo and Colorni (1999). The Ant System algorithm joined with local search obtained solutions of comparable quality to the Greedy Randomised Adaptive Search Procedure (GRASP) with improved processing time (McCallum, 2005).

A number of algorithmic improvements have been proposed to improve Ant System performance. All these improvements have in common that they introduce a form of elitism which is able to direct the search more strongly towards the best tours. In addition, because the major concern in AS was the treatment of pheromone trail intensities on arcs which may leads to early stagnation. The following sections are a description of the main extensions of the AS algorithm.

5.3.1 Ant-Q

The first main extension to the Ant System was found in Gambardella and Dorigo (1995), with an adaptation in Taillard and Gambardella (1997). It was a mixture of the Ant System and Reinforcement Learning. Let $AQ(r, s)$, read Ant-Q-value, be a positive real value connected to the arc (r, s). It is the Ant-Q equivalent of Q-learning Q-values, and is aimed to indicate how useful it is to make move from city r to city s. Let $HE(r, s)$ be a heuristic value connected to node (r, s) which allows a heuristic evaluation of which moves are better (in the TSP, the inverse of the distance is usually chosen).

Assume that k is an agent whose task is to construct a tour in the TSP: visit all the cities and return to the starting city. Connected to k, there is a list $J_k(r)$ of cities not visited, where r is the current city. This list uses a kind of memory to prevent transitions to previously visited

cities and to force the ant to build legal tours. An ant *k* placed in city *r* moves to city *s* using the equation (5.4), called the action choice rule (or state transition rule):

$$s = \begin{cases} \arg \max_{u \in j_k(r)} \left\{ [AQ(r,u)]^{\delta} \times [HE(r,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \qquad (5.4)$$

where δ and β are parameters which bias the relative importance of the learned *AQ*-values and the heuristic values, q is a value chosen randomly with uniform probability in [0, 1], q_0 $(0 \le q_0 \le 1)$ is a parameter such that the higher q_0, the smaller the probability of making a random choice (good values of q_0 tend to be close to 1(Stützle et al., 2010)) and *S* is a random variable selected according to a probability distribution given by a function of the *AQ*(*r, u*) and *HE*(*r, u*) values, with $u \in J_k(r)$. These *AQ*-values were updated by equation (5.5).

$$AQ(r,s) = (1-\rho) \times AQ(r,s) + \rho(\Delta AQ(r,s) + \gamma \times \max_{z \in J_k} AQ(s,z)) \qquad (5.5)$$

where J_k is the list of cities still not visited by ant *k* and γ is the discount rate (horizon) which is a value in the range [0, 1]. Gambardella and Dorigo (1995) suggested γ = 0.3. The idea of local trail updates and global trail updates was introduced in their work. The purpose of the former was to try and diversify the pheromone matrix, as using global trail updates alone was found to converge the matrix too early. The other rules were all very similar to those of the Ant System.

The most interesting contribution of the work of Gambardella and Dorigo (1995) was the definition of the λ-branching factor. The λ -branching factor calculates an estimate of the size of the search space being focused on by the algorithm at any point in time. Its definition is based on the following concept: if for a given node *i* the concentration of the pheromone trail on approximately all the arcs exiting from it becomes very limited, the freedom of choice for

expanding partial solutions from that node is very small. As a result, if this condition occurs concurrently for all nodes of the graph, the part of the search space that is effectively searched by the ants becomes very small. The branching factor for node *i* is defined as follows: if τ_i^{max} is the maximum and τ_i^{min} is the minimum trail intensity on arcs exiting from node *i*, the λ-branching factor is given by the number of arcs exiting from *i* which have trail intensity as given in equation (5.6). The average-branching factor is the average of the branching factors of all nodes and gives an indication of the size of the search space explored by the ants. For example, if the average-branching factor is very close to 3, it means that, on average, three arcs incident to each node are likely to be chosen. The parameter λ acts as a threshold level, which is set to a value in the range $0 < \lambda < 1$. To eliminate the influence of different settings for λ, Ridge (2007) suggested that a fixed value ($\lambda = 0.05$) may be used.

$$\tau_{ij} > \tau_i^{min} + \lambda(\tau_i^{max} - \tau_i^{min}) \qquad (5.6)$$

where τ_i^{max} and τ_i^{min} are the extreme pheromone intensity values in the pheromone matrix (τ), *i* is a node and *j* is the arc connects between two nodes.

In Gambardella and Dorigo (1995), Ant-Q was compared to the Ant System using the TSP and was generally found to be better in terms of the mean solution found, but the best results obtained in number of iterations for each algorithm were the same. Mariano and Morales (1999) used Ant-Q in the design of water distribution irrigation networks, a complex real-world problem. Gambardella and Dorigo (1995) used Ant-Q to successfully solve TSP (Oliver 30) that has been solved by genetic algorithm. They found that the mean (424.44) and the standard deviation (0.46) achieved by the Ant-Q were better than the mean (425.44) and the standard deviation (0.51) achieved by GA.

5.3.2 Ant colony system (ACS)

The ACS was introduced by Dorigo and Gambardella (1996 and 1997) to improve the performance of the AS. A number of aspects of the Ant-Q and Ant System were fused to create an improved version denoted as the ACS algorithm. The concept of the algorithm is based on a number of changes made to the original ant system. The aim was to find a balance between exploration and exploitation to avoid early convergence. There are three main differences between the ACS and AS: tour construction, local pheromone trail updates and global pheromone trail updates. The three main differences are:

Tour construction: in the ACS, ants use a different decision rule, called the *pseudo-random-proportional* rule, in which an ant *k* on node *i* chooses the node $j \in N_i^k$ to move to as shown in equation (5.7):

$$j = \begin{cases} \arg\max_{j \in N_i^k} \{(\tau_{ij})^{\alpha} \times (\eta_{ij})^{\beta}\} & \text{if } q \le q_0 \\ J & \text{otherwise} \end{cases} \qquad (5.7)$$

where q is a random number uniformly distributed over [0, 1], and $q_0 \in [0,1]$ is a tuneable parameter to modulate the degree of exploration. Thus, the best possible move, as indicated by the pheromone trail and the heuristic information, is made with probability $0 < q_0 < 1$ (exploitation); with probability $1-q_0$ a move is made based on the random variable J with distribution given by equation 5.1 (biased exploration). Dorigo and Gambardella (1996 and 1997) explained that the decision rule has a two-fold purpose: when $q \le q_0$, the decision rule exploits the information available on the problem, while when $q > q_0$, it performs a biased exploration.

Global pheromone trail update: in the ACS, only the global best ant is allowed to add pheromone after each iteration. Thus, the update is implemented as given by equation (5.8):

$$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(t) + \rho \Delta \tau_{ij}^{gb}(t) \qquad (5.8)$$

where *gb* is the global-best ant.

$$\Delta \tau_{ij}(t) = \frac{1}{L_{best}}$$

where L_{best} is the best solution so far.

Local pheromone update: in addition to the global pheromone trail updating rule, ants use the local update rule that they apply immediately after having crossed an arc (*i, j*) during tour construction, as shown in equations (5.9).

$$\tau_{ij} = (1-\rho) \times \tau_{ij} + \rho \Delta \tau_0 \qquad (5.9)$$

where $\rho \in$ (0, 1] is a parameter governing pheromone decay and τ_o is the parameter specifying the initial value of the pheromone matrix.

The ACS has been used to solve the Assembly Line Balancing problem joined with local search to improve solutions (McCallum, 2005). The algorithm outperformed many metaheuristics. Silva et al. (2002) applied the ACS algorithm to Logistic Process Optimising. They used a simulator to investigate how the parameters influenced the quality of the schedules produced. These experiments were based on real-world data. The results achieved by the ACS proved to be better than the previously used method.

5.3.3 Rank-based ant system (AS_{rank})

The rank-based Ant System is a further improvement on the Ant System (Bullnheimer et al., 1997). In this algorithm, the global-best tour is used to update the pheromone trails. Furthermore, a number of the best ants of the current iteration are allowed to update the pheromone trail. For this aim, the ants are sorted by tour length ($f(s_1) \leq f(s_2) \leq \ldots.. \leq f(s_m)$), and the quantity of pheromone an ant may deposit is weighted according to the rank *r* of the ant. In each iteration, only the (*w*-1) best ranked ants and the ant that produced the best-so-far

tour are allowed to lay some quantity of pheromone. The global best solution gives the strongest feedback of weight *w*. The *r*th best ant of the current iteration is allowed to drop maximum amount of pheromone {0, *w*-*r*}. The notation is given below and is followed by the altered AS_{rank} pheromone update rule as given by equation (5.10):

- *r* is the rank of an ant by fitness, for instance *r*=1 points to the ant with rank of 1.
- w_{best} is the fitness of the best ant found so far.
- ω is the number of ants to rank (*w*-1).
- $\Delta\tau_{ij}^{r}$ is the increase of trail intensity on an edge (*i* ,*j*) caused by the *r*-th best ant.
- L_r is the tour length of the *r*-th best ant.
- $\Delta\tau_{ij}^{r}$ is the increase of the trail intensity on an edge (*i* ,*j*) caused by the elitist ants.
- *w* is the number of elitist ants. Elitist ants are those that are allowed to imprint on the pheromone matrix.
- L^{gb} is the tour length of best solution found.
- *Q* measures the influence of the new information relative to the influence of the initial trail level.

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{r=1}^{w-1}\Delta\tau_{ij}^{r}(t) + w\times\Delta\tau_{ij}^{gb}(t) \qquad (5.10)$$

where $\Delta\tau_{ij}^{r}(t) = \begin{cases} \{w_{best} - r\}\times Q/L_r & \text{if the r - th best ant travel on edge } (i, j) \\ 0 & \text{otherwise} \end{cases}$

$$\Delta\tau_{ij}^{gb}(t) = \begin{cases} w\dfrac{Q}{L^{gb}} & \text{if edge } (i, j) \text{ is part of the best solution found} \\ 0 & \text{otherwise} \end{cases}$$

Empirical results in Bullnheimer et al. (1997) suggest that AS_{rank} performs better than AS. The average deviation from the optimal solution obtained by the AS_{rank} was 1.81%, whereas the average deviation from the optimal solution obtained by the AS was 2.57%. In Bullnheimer et al. (1999) AS_{rank} was compared to AS, to a genetic algorithm and to simulated annealing. It was found that for the larger TSP cases (the largest case with 132 cities); the performance of the AS_{rank} was found to be superior to the genetic algorithm and the simulated annealing procedure.

5.3.4 Max-min ant system (MMAS)

The max-min ant system was introduced by Stützle and Hoos (1997, 1998 and 2000). Its concept is based on using elitism to introduce exploitation to the original ant system and to avoid early stagnation of the search. Stagnation will be described in this section. The MMAS was initially applied to the TSP and QAP and enjoyed greater success than the original Ant System algorithm (Stützle, 1997, Stützle and Hoos, 1998). The MMAS differs in three key aspects from the AS:

(i) Only one single ant is allowed to reinforce pheromone trails after each iteration. This ant may be the one which found the best solution in the current iteration (*iteration-best* ant) or the one which found the best solution from the beginning of the trail (*global-best* ant).

(ii) Values for pheromone trails are limited to an interval $[\tau_{min}, \tau_{max}]$ in order to avoid early stagnation of the search, hence the name max-min.

(iii) Deliberately initialising the pheromone trails τ_0 to $\tau_{\max}$ in this way achieves a higher exploration of solutions at the start of the algorithm.

(iv) Pheromone trails are reinitialized when the system approaches are stagnated or when no improved tour has been generated for a certain number of consecutive iterations.

In MMAS, only one ant is permitted to update the pheromone matrix after every iteration. This ant provides either the *global best ant* (*gb*) or the *local best ant* (*lb*) solution. The modified pheromone trail update rule is given by equation (5.11).

$$\tau_{ij}(t+1) = \rho \times \tau(t) + \Delta\tau_{ij}^{best}(t) \qquad (5.11)$$

where $\Delta\tau_{ij}^{best}(t) = \frac{1}{f(s_{best})}$ and s_{best} may be either the ant with the iteration-best tour or the one with the global-best tour and ρ is the pheromone evaporation rate in order to avoid unlimited accumulation of the trail; the value of ρ should be $(0 < \rho \leq 1)$.

The use of only one solution, either the global best (*gb*) or local best ant (*lb*), for the pheromone update is the most important means of search exploration in the MMAS. With this choice, solution elements which frequently occur in the best found solutions receive a large reinforcement. In the MMAS, when using only the global-best ant, the search may concentrate too quickly around this early global-best solution and the exploration of possibly better tours is limited. Consequently, the danger of getting trapped in poor quality solutions is higher. On the other hand, using the local best ant favours the exploration of possibly better tours since, especially in the starting phase of the algorithm, the local best ant will differ considerably and early mistakes are more easily avoided. The local and global search is also used in PSO algorithm. The PSO actually uses both aspects of cooperation and competition among the individuals in the population, which means it combines local and global search to reach the global optima, see section 4.3.8. In addition, intermediate approaches can be applied, such as choosing by default the local best ant to update the pheromone trails and using the global best tour only every fixed number of iterations. This mixed strategy which is based on local-best ant and global-best ant for updating pheromone trails will be used in the AOIS problem. The aim of this strategy is to obtain stronger exploration of the search space early in the search and stronger exploitation of the overall best solution later in the run.

The MMAS and ACS both exploit the best solutions by using only a single ant in the pheromone trail update. However, an important difference between them is the different interpretation of exploitation and exploration. Exploitation in the ACS is mainly interpreted as choosing a high parameter value for q_0, see equation (5.7). In this way, the accumulated knowledge on the problem is exploited by constructing a solution that can be interpreted as a slight modification of the best solution found so far. Exploration in the ACS is obtained using a biased random move according to equation (5.1) with a probability of $(1-q_0)$ (Stützle, 1998b).

On the other hand, in MMAS, exploitation is mainly interpreted as choosing one single ant, either the local-best ant or the global-best ant, for the pheromone update. Jointly, with a rather high parameter value for ρ, this will slowly shift the probability distribution given by equation (5.1) towards solution components (arcs) which have been shown to be contained in the best solutions. Exploration in the MMAS is derived from explicit pheromone trail limits which aim to increase the solution exploration of the algorithm.

One of the major important features in the MMAS is avoiding of stagnation. Stagnation is the situation in which all ants follow the same path and construct the same tour, which in general is highly suboptimal (Dorigo et al., 1996). In other words, stagnation of the search occurs, for example, in the following situation. If the amount of pheromone on only one arc incident from a node is very high compared to the other arcs, this arc has a high probability of always being selected using equation (5.1). If such a situation occurs at all nodes, the tour corresponding to the best one found so far will be constructed by most ants and the search for better solutions stagnates. To avoid this situation, one possibility is to limit the pheromone trail. This goal can be achieved by limiting the values for pheromone trails to an interval

[τ_{min}, τ_{max}]. After updating pheromone at the end of each iteration, the pheromone trails τ_{ij} on all arcs are reinforced to be within these limits, as shown in equation (5.12).

$$\tau_{\min} \leq \tau_{ij}(t) \leq \tau_{\max} \tag{5.12}$$

The maximum limit of the trail intensity for the MMAS is calculated as shown in equation (5.13) (Stützle and Hoos, 2000).

$$\tau_{\max} = \frac{1}{1-\rho} \frac{1}{f(s_{\text{opt}})} \tag{5.13}$$

where $\tau_{\max}$ is the maximal pheromone trail and $f(s_{\text{opt}})$ is the optimal solution value for the problem. Clearly, the optimal solution value is not known before the run, and $f(s_{opt})$ is used as an estimate of that value and then is adapted during the running of the algorithm.

The lower limit of the trail intensity shown in equation (5.14) is calculated with consideration of a number of assumptions. First, it was assumed that the best tours would be found just before stagnation and that, more importantly, better tours were to be found near to the best tours. In such a case, the probability that the best tour found is constructed in one iteration is significantly higher than zero. Through experimentation this property has been shown to be reasonable for TSP benchmark problems (Stützle and Hoos, 2000, Stützle, 1998b). The other assumption was that the main influence on tour construction was the relationship between the upper and lower trail limits.

$$\tau_{\min} = \frac{\tau_{\max} \times (1 - p^{dec})}{avg \times p^{dec}}, \quad p^{dec} = \sqrt[n-1]{p_{best}} \tag{5.14}$$

where $\tau_{\min}$ represents the lower limit for the pheromone trail strength and *avg* is the average number of available options the ant has to choose from at any decision point. The best solution found is constructed with a probability p_{best} which is significantly higher than 0. The optimal value for probability p_{best} is 0.05 (Stützle and Hoos, 2000, Ridge, 2007). Other

authors, such as Ridge (2007), calculated the lower trail limit $\tau_{\min} = \frac{\tau_{\max}}{2n}$, where *n* is the problem size (e.g. number of cities). Stützle (1998a) set the lower pheromone trail limit to $\tau_{\min} = \frac{\tau_{\max}}{5}$ for the Flow Shop Problem.

Search progress in the MMAS implies a specific interpretation, by the particular way in which the pheromone trails are initialised. The pheromone trail reduces due to evaporation to $\tau_{ij}(t+1) = \rho \times \tau_{ij}(t)$ after each iteration of the algorithm. Only the pheromone trails of arcs participating in the best tours increase their pheromone trail or keep them at the upper trail limit, because only the best ant is allowed to update the pheromone trail. Arcs which do not obtain regular reinforcement to their pheromone trail will be maintained lower and be chosen more rarely by the ants. In this sense, errors made in the past are avoided in the MMAS. An error is associated with choosing arcs that lead to fairly bad tours; these are denoted as *poor* arcs. Thus, the pheromone trail on *poor* arcs decreases gradually and only *good* arcs keep a higher level of pheromone. These good arcs are then combined by the probabilistic tour creation to generate improved tours (Stützle and Hoos, 2000).

Stützle and Dorigo (1999) applied the MMAS to the TSP and their results compared to Iterated Local Search. The MMAS found a better solution than the other algorithms used for 77% of the problems on an average run. The MMAS was used in Stützle (1998b) to attack the Flow Shop Problem. The algorithm outperformed a number of other methods such as Simulated Annealing and Multiple Descent. The experimental results in Stützle and Hoos (2000) demonstrate that the MMAS achieves strongly improved performance compared to the AS and to other improved versions of the AS for the TSP; moreover, the MMAS is among the best available algorithms for the QAP. The MMAS was applied to the University Timetabling Problem by Socha et al. (2002 and 2003). The paper showed that the algorithm performed better at a set of problem instances than an algorithm using the local search with

random starting solutions. Ning et al. (2010) applied the MMAS to the construction site layout planning (CSLP) problem. CSLP problem is a dynamic multi-objective optimisation problem as there are different facilities employed in the different phases of a construction project. The CSLP problem was modelled as a QAP under two objective functions of minimising the representative score of safety/environmental concerns and the total handling cost of interaction flows between the facilities associated with the construction site layout. The experimental results show that the safety level is improved and the construction cost is reduced.

- **Features of MMAS algorithm**

The following are the important features for the MMAS algorithm:

1. One of the major important features in the MMAS algorithm is avoiding too early stagnation.
2. By using only one single ant in the MMAS algorithm the pheromone trail update, the best solutions can be better exploited.
3. By using the adaptive memory allows the previously visited workstations to be kept by means of the pheromone trail matrix, which is used to influence the construction of new better solutions.
4. The heuristic information helps to find acceptable solution in the early stages of the search process.
5. The collective interaction of a population of ants leads to increase the exploration of the search space.
6. MMAS algorithm starts with initial high pheromone trial which is leads to increase exploration of the search space.

5.4 Local search mechanisms

Research in ant colony optimisation has shown that for applications on combinatorial optimisation problems, the best results are obtained if the ants are enhanced by additional capabilities. The local search is part of the *DaemonActions* of the ACO algorithm as shown in the ACO framework in section 5.2.2. In many of the most efficient implementations of ACO algorithms, ants may apply local search to improve the solutions they have constructed (Dorigo and Gambardella, 1997, Stützle and Hoos, 2000). Therefore, many researchers have developed local search for ACO and have applied these methods to a range of different problems such as the redundancy allocation problem (Liang and Smith, 2004), the inter-cell layout problem in cellular manufacturing (Solimanpur et al., 2004), single row layout in flexible manufacturing systems (Solimanpur et al., 2005), image pre-processing (Laptik and Navakauskas, 2009) and construction site layout planning (Ning et al., 2010). The reason for adding local search algorithms to ACO is to enhance performance and to yield high quality solutions, such that near-optimal solutions can be found. Advantages and disadvantages of using local search can be explained as following:

- **Advantages of local search**

1. Cost of generating neighbouring solutions: typically, for generating a neighbouring solution the computational complexity is much lower than generating a new solution from scratch. In addition, for evaluating a neighbouring solution, it often does not need to generate it explicitly at all.
2. Cost of evaluating neighbouring solutions: typicallyΔ-evaluation can be done in a computational cost that is much less than computing solution cost from scratch (Stützle, 2003).

- **Disadvantages of local search**

1. Iterative improvement may take exponential time in the worst case but usually this occurs only rarely and for few problems.
2. Problem of local optimality.

The particular local search used is almost completely problem-dependent, but the important idea is that a solution to the problem in hand has an identifiable solution neighbourhood. Usually, the neighbourhood of a solution can be defined as all those solutions which may be different from the original solution by a single ‘‘step’’. The most well-known local search algorithm, called iterative improvement, first builds an initial solution by some means (possibly creating one at random). The algorithm then checks through some or all of the neighbours of the initial solution looking for better solution. If an improved solution is obtained, then the current solution is replaced with it and the process repeats until no further improvement can be found. A drawback of this algorithm is that it may stop at poor quality local optima. As a result, possibilities have to be devised to improve its performance. One would be to increase the size of the neighbourhood used in the local search algorithm. Clearly, there is a higher possibility to obtain an improved solution, but it also takes longer time to assess the neighbouring solutions. This leads to this approach being impractical as the neighbourhoods increase. Another possibility is to allow the local algorithm to generate a new random solution. However, the search space normally contains a massive number of local optima. As a result, this approach becomes increasingly ineffective as the problem becomes complex.

Several applicable improvements of local search methods have been suggested. The aim is to avoid these drawbacks of iterative improvement methods. The improvement of the local search methods is based on either by accepting worse solutions, hence allowing the local search to escape from local optima, or by creating good starting solutions for local search

algorithms in a more intelligent way (such as ACO) than just providing random initial solutions (Solimanpur et al., 2004). Moreover, since the improvement algorithms start from bad solutions, the computation time required by improvement algorithms would be more. Besides, ant algorithms are population-based adaptive metaheuristics that are able to construct relatively good solutions and therefore their integration with a local search mechanism may result in optimum or near optimum solutions. Owing to the fact that the quality of initial solutions created by ant algorithms is good, the integrated local search mechanism needs only a few iterations to enrich these solutions to their local optimum resulting in reasonable computation time. In addition, to yield a further reduction in run-time and to focus the local search around the part where potentially improve can be found *don't look bits* is used (Ferreira et al., 2012). This prevents cycling, and also helps to promote a diversified coverage of the search space.

To apply iterative improvement algorithms, two commonly recognised approaches are first ascent and best ascent. In the first method, the first neighbour which is an improvement is selected, whereas the second method searches through all potential neighbouring solutions and then selects the one which offers the greatest improvement. In TSP, the 2-opt neighbourhood is defined as a neighbour if it differs by at most two arcs. This can be generalised to deal with *k* arcs. An example of a solution and an improved neighbour is given in Figure 5.5.

The idea of using a mechanism to generate initial solutions which are improved by a subsequent local search is also applied in other nature-inspired algorithms such as simulated annealing and genetic algorithms. The most recent of these nature-inspired algorithms is ant colony optimisation. In the ACO algorithm, the search process can be used to identify promising regions of the search space with high quality solutions. This can be done by using

pheromone trails as an adaptive memory of solution components which have been part of the best local minima found so far.

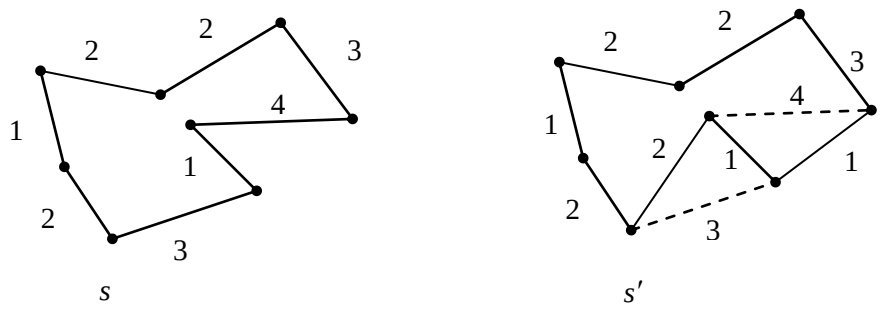


Figure 5.5: Illustration of a 2-opt neighbour in the TSP. (s' is an improved neighbour of s because it varies by two arcs, and its tour length is 4-four less than s) (Ritchie, 2003).

These solution components are combined in subsequent iterations by a stochastic construction mechanism, which is biased by the pheromone trails and local heuristic information. A further advantage of using a constructive algorithm such as the MMAS is that by creating good initial solutions, the subsequent local search requires fewer steps to arrive at a local optimum. However, based on the previous literature review in chapter 2, none of the surveyed methods in the AOIS problem used local search to improve their performance. Local search methods for the AOIS problem will be developed to improve the performance of the MMAS algorithm and will be introduced in the next chapter.

5.5 Fitness Landscape

It should be noted that none of the studies surveyed in chapter 2 investigated the fitness landscape for the AOIS problem. As a result, this section will provide further information about understanding the fitness landscape of the AOIS problem. In the AOIS problem, it is clear that a process including n serial stages offers 2^n possible inspection combinations. As the number of stages increases, complete enumeration of all combinations becomes

prohibitive. Therefore, the application of metaheuristic methods is essential for developing a tractable solution algorithm, as these methods need limited computational effort while yielding a nearly optimal solution. The objective of this research is to allocate limited inspection stations in a serial multistage manufacturing process. If this is done, the total manufacturing cost of a product can be reduced without affecting the quality of the product. The total cost is defined as the sum of the costs of production and inspection, and internal and external failure costs. In the AOIS problem, it is possible that not all locations of the inspection stations are economically equivalent. Because of the difference in cost structure in inspection places and process characteristics, some combinations of inspection plans may prove to be economically preferable to others. This is because the product is processed through different operations by different machines. These machines are different from each other in terms of their characteristics such as operation cost, scrap cost, rework cost, inspection cost and defective rate. The differences in process characteristics lead to differences in the cost structure of inspection plans. Understanding the geometry of the landscape for the AOIS problem may help in choosing the appropriate algorithm to solve the AOIS problem.

Fitness landscape was first introduced by Wright (1932) in a study of evolutionary theory. Intuitively, the fitness landscape can be imagined as a mountainous region with hills, craters, and valleys. The local search algorithm can be pictured as a wanderer performing a biased walk in this landscape. His goal is to find the lowest point (in the case of minimisation problems such as the AOIS problem) in this landscape. It is obvious that the task for the wanderer strongly depends on the ruggedness of the landscape, the distribution of the valleys and craters and the local minima in the search space, and the overall number of local minima (Yamada, 2003). It is widely agreed that the performance of metaheuristics depends strongly on the characteristics of the underlying search space. The difficulty of searching in a given

problem search space is related to the structure of the fitness landscape (Smith et al., 2002).The fitness landscape of a combinatorial optimisation problem is defined by a triplet (Ω, N, f) where Ω is the set of solutions called the search space, *N* is a neighbourhood function and *f* is an evaluation function (Marmion et al., 2012). With this definition, the notion of the neighbourhood between solutions takes a significant place in the resolution of combinatorial problems. This notion is used in local searches as the application of a defined operator. Thus, an analysis of such a fitness landscape will be helpful in order to understand the structure of a problem from a local search point of view.

The fitness landscape determines the shape of the search space as encountered by a local search algorithm. Marmion et al. (2012) stated that a fitness landscape could be seen in 2D or 3D, as a topographic representation of the problem where the relief is given by the difference of the fitness between neighbouring solutions. The link between landscape and search algorithm is given by the neighbourhood search (*NS*) operators used in the algorithm. Because these operators generate new points in the search space relative to a given point, the distance between two solutions is equal to the minimum number of required applications of the operator to move from the first one to the second one (Yamada, 2003;Marmion et al., 2012).

Consider local search algorithms like the ACO: if the average cost difference between neighbouring solutions is, on average, small, the landscape will be well suited to a local search algorithm; if the average cost difference between neighbouring solutions is high, the landscape is rather rugged and may contain many local minima and be badly suited to a local search algorithm (Angel and Zissimopoulos, 2000).The distribution of local minima and their relative location with respect to global optima is an important criterion for the effectiveness of adaptive algorithms like ACO algorithms. For analysing this aspect of the fitness

landscape, the correlation between solution fitness and the distance to optimal solutions has been studied by Weinberger (1990), Stadler (1996) and Merz and Freisleben (2000).

5.5.1 A distance measure

The fitness landscape analysis relays on a distance metric for AOIS solutions. To measure the difference between two permutation inspection plans s and s', an appropriately defined distance is required. The distance between two points (solutions) can be defined as the minimum number of elementary move operators which have to be applied to transform one permutation into the other permutation. In the case of the AOIS problem, the swap operator is used. The swap operator is widely used for permutation problems (Marmion et al., 2012).The *swap distance* is based on the swap move which exchanges a pair of inspection stations π_i and π_j with $i \neq j$. The calculation of the exact swap distance between two permutations is nontrivial (Czogallaand Fink, 2012). In order to reduce computational complexity, a path in the swap neighbourhood may be calculated by position-wise comparison of the two parent permutations π and π'. If an inspection station in π is not in the same position as in π' it is swapped to the correct position and the move is stored.

5.5.2 Fitness distance correlation

Fitness distance correlation (FDC) was first proposed by Jones and Forrest (1995) as a measure of problem difficulty for evolutionary methods. It was first used to analyse the hardness of a problem for a genetic algorithm, but it also gives very useful hints on the effectiveness of adaptive algorithms which use discontinuous trajectories, such as the ACO algorithm (Stützle, 2000). An FDC analysis has been conducted for various combinatorial optimisation problems, including travelling salesman problems (Boese, 1995), flow-shop scheduling problems (Reeves, 1998), graph partitioning problems (Merz and Freisleben, 1998) and timetabling problems (Ochoa et al., 2009).The FDC is important for the

interpretation of search performance of discontinuous metaheuristics. This is because the notion of search space region is tightly coupled with the notion of distance between solutions. The task of adaptive restart algorithms like the ant colony optimisation is to guide the search towards regions of the search space containing high quality solutions and, possibly, the optimum. Recall that the most important guiding mechanism of these metaheuristics is the objective function value of solutions. This guidance mechanism relies on the intuition that the better a solution is, the more likely it is that even better solutions will be found close to it. In particular, the fitness distance correlation describes the relationship between the fitness (cost) of solutions and their distance to best-known solutions or optimum solutions.

In other words the FDC is the correlation between the quality of a solution and its distance to an optimal solution. The FDC states how closely fitness and distance to an optimal solution are related. Hence, if a problem shows a high FDC, algorithms combining adaptive solution generation and local search may be expected to perform well. For ACO algorithms this is the case because the most important guidance mechanism of ACO algorithms is the solution quality of the solutions constructed by the ants; the better a solution, the more its solution components will be reinforced. Yet if no such correlation exists, or, even worse, if cost and distance are negatively correlated, the fitness gives only little or no guidance towards better solutions and the ACO algorithm may perform poorly on such problems.

In a problem instance with high FDC, good solutions tend to be tightly clustered or, equivalently, to share many solutions attributes. Consequently, an adaptive search algorithm should be able to exploit these similarities during a search. The easiest way to measure the extent to which the fitness function values are correlated with distance to the optimum is to examine a problem with known optima, take a sample of individuals and compute the FDC, given the set of (fitness, distance) pairs. Formally, given a set $F=\{f_1,f_2,...,f_n\}$ of individual

fitnesses and a corresponding set $D=\{d_1,d_2,...,d_n\}$ of the n distances to an optimal solution or to the nearest global optimum, the FDC is defined by Jones and Forrest (1995)as:

$$FDC = \frac{C_{FD}}{\sigma_F \sigma_D} \tag{5.16}$$

where σ_F and σ_D are the standard deviations of F and D, respectively and are defined by equation (5.17).

$$\sigma_F = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})^2}\,, \qquad \sigma_D = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \bar{d})^2} \tag{5.17}$$

C_{FD} is the covariance of F and D. The covariance C_{FD} is defined by equation (5.18):

$$C_{FD} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d}) \tag{5.18}$$

where $\bar{f}$ and $\bar{d}$ are the means of F and D respectively.

High *FDC* values indicate that fitness and distance to the optimum are related, and that the search promises to be relatively easy for the technique being used, since there is a path to the optimum via solutions with increasing fitness. For minimisation problems, the ideal fitness function will have *FDC*=1 (Jones and Forrest, 1995). Similarly, in the AOIS problem, a high positive correlation between the solution cost and the distance to the optimum indicates that the better the solution, the closer the algorithm gets, on average, to the optimum. If no such correlation exists or it is very weak, the fitness gives only little guidance towards better solutions. As discussed by Jones and Forrest (1995), Jaszkiewicz and Kominek (2003) and Muller and Sbalzarini (2011), the coefficient FDC is expected to be near 1 for a globally convex single-funnel or a big valley structure and around 0 for needle-in-a-haystack problems and problems without any global structure. A negative value of FDC indicates a “deceiving” and misleading landscape.

An FDC analysis for the AOIS problem search space using the PSO and MMAS algorithms being studied, which examine all possible allocations of the inspection stations problem, is plotted. Each algorithm is combined with a local search method (swap). These two algorithms are used here because they have the most desirable characteristics of all the methods described in chapter 4. The aim is to find an algorithm suitable for tackling the AOIS problem. Often, a fitness distance plot is made to gain insight into the structure of the landscape, in addition to calculating the correlation coefficient (Ochoa et al., 2009). The fitness distance plot is done by plotting the fitness of points in the search space against their distance to the optimum or best-known solution. This type of analysis, often called *fitness distance analysis*, can be used to investigate not only the correlation between arbitrary points in the search space, but also the distribution of local optima within the search space.

Figure 5.6 shows scatter plots of percentage deviation from the optimum versus the distance from the optimal solution for 15 workstations for the AOIS problem. The number of feasible solutions for this number of workstations is 2^{15}=32,768. For this number of workstations the optimal solution is known. Each point gives the distance to the optimum (x-axis) and the solution quality as the percentage deviation from the optimum (y-axis). The plots show a strong correlation between the solution quality and the distance to the optimum, which can be seen be by the fact that better local optima tend to be closer to the optimal solution.

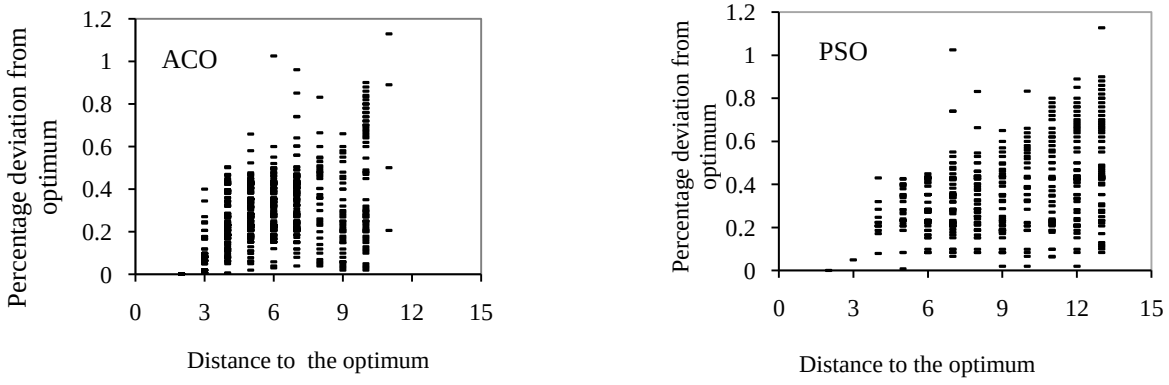


Figure 5.6: Fitness distance scatter plots for 15 workstations for the AOIS problem

It can be seen that for the ACO algorithm that there are local optima which are much closer to the optimum solution than there are for the PSO algorithm. The results for the FDC analysis are given in Table 5.1, which shows the average percentage deviation from the optimum, the average distance to the optimum, the respective ratios to the maximum possible distance, and the correlation coefficients. It can be concluded that, on average, the better the solution quality, the closer a solution is to an optimal solution in the AOIS problem. Comparing the results of the FDC analysis indicates the potential usefulness of the ACO approach to the AOIS problem. The FDC for the ACO algorithm is 0.831, which indicates a strong correlation between the solution quality and the distance to the optimum. Similarly, the FDC obtained by the PSO is 0.57, which also shows there is a good correlation between the solution quality and the distance to the global optimum. The studies introduced by Jaszkiewicz and Kominek (2003), Grahl et al. (2007) and Muller and Sbalzarini (2011) showed that high FDC coefficients are an indicator for the presence of a big valley structure. A big valley structure means that local optima tend to be relatively close to each other and to the global optimum. In a big valley structure, the local search can potentially drive the search towards the neighbourhood of an optimal solution (Grahl et al., 2007).The results indicate that the landscape produced by the ACO has deeper valleys. Thus it seems that the ACO should move to a local optimum with a better quality more easily than the PSO does.

Table 5.1: Results of the FDC analysis for 15 workstations for the AOIS problem

Method	Average %	$avg_{d\text{-}opt}$	$avg_{d\text{-}opt}/n$	FDC
PSO	1.72	8	0.53	0.57
ACO	0.166	4	0.26	0.831

Average %: the average percentage deviation from the optimum, $avg_{d\text{-}opt}$: the average distance to the optimum, $avg_{d\text{-}opt}/n$: the ratio between average distance to the optimum ($avg_{d\text{-}opt/n}$) and the problem size (n: is the number of workstations), FDC: fitness distance correlation coefficient.

Further analysis on different AOIS problems using the ACO algorithm is presented in Table 5.2. The solutions are interconnected differently in the search space according to the

neighbourhood function. Pair-wise distances of a population of solutions are meaningful for characterising the search space. The usual statistics of all these distances are computed. Thus, the comparison of distances between solutions of the local optima helps to characterise whether the solutions are identically distributed and dispersed, according to whether or not they are local optima. The distribution of solutions should help us to understand if they are close or not, and if local optima are in the same part of the search space (Marmion et al., 2012). The indicators standing for mean, standard deviation, minimum, maximum and quartile values are then computed to estimate the width of the search space. Table 5.2 presents the average distance to the optimum, the respective ratios to the maximum possible distance, the minimum, the median (Med), the first quartile value (Q_1), the third quartile value (Q_3), the maximum and the FDC. It is concluded that, on average, the better the solution quality the closer a solution is to an optimal solution in the AOIS problem. The quartiles (Q1 and Q3) and median show that the good solutions are very concentrated, and that the distances between the solutions are homogenous and small. As a result the algorithm stays in a limited region of the search space. The FDC for the ACO algorithm indicates a strong correlation between the solution quality and the distance to the optimum for each problem considered.

Table 5.2: Results of the FDC analysis for different AOIS problems using the ACO

Problem size	$Avg_{d\text{-}opt}$	$Avg_{d\text{-}opt}/n$	Min	Med	Q_1	Q_3	Max	FDC
15	4	0.26	3	5	4	6	10	0.831
16	6	0.375	4	6	5	7	12	0.78
18	8	0.44	7	9	8	10	14	0.79
20	11	0.55	9	11	9	12	16	0.76

$avg_{d\text{-}opt}$: the average distance to the optimum, $avg_{d\text{-}opt}/n$: the ratio between average distance to the optimum ($avg_{d\text{-}opt/n}$) and the problem size, Min :minimum distance, Med: median, Q_1:first quartile, Q_3: third quartile, Max: maximum distance and FDC: fitness distance correlation coefficient.

5.6 Conclusion

Ant colony optimisation has been described followed by the Ant System and its extension versions. Different ant colony versions were studied, leading to the MMAS algorithm being proposed to tackle the AOIS problem. The MMAS has been shown to perform significantly better than many optimisation methods, especially with complex problems. It was found that, in the AOIS problem, there is a need to construct heuristic information which aids the performance of the MMAS. Research has shown that when the MMAS is applied to combinatorial optimisation problems, the performance of ant algorithms is best when joined with local search methods. The local search in the MMAS algorithm aims to improve the solutions constructed by the ants. In the AOIS problem, a complete solution (inspection plan) is achieved when an ant visits workstations in the serial multistage manufacturing process. As there are a limited number of inspection stations available. Therefore, the number of workstation to be visited by an ant is restricted by the number of inspection stations available. The advantage of using the MMAS is the adaptive memory that allows the previously visited workstations to be kept by means of the pheromone trail matrix, which is used to influence the construction of new solutions. In particular, this matrix will include the paths (inspection positions) that ants have visited. Subsequently, the path which has the lowest cost will be used more frequently by subsequent ants. In addition, the MMAS has the ability to avoid early stagnation using pheromone trail bounds. Also, by using only one single ant in the pheromone trail update, the best solutions can be better exploited. The fitness distance correlation (FDC) for the AOIS problem using the MMAS algorithm indicates strong correlation between the solution quality and the distance to the optimum. This indicates that the MMAS algorithm is well suited to the AOIS problem. The application of the MMAS algorithm on the AOIS problem will be described in the next chapter.

Chapter 6
Max-min ant system for the allocation of inspection stations

In this chapter, the MMAS algorithm is evaluated against a new type of problems known as the allocation of inspection stations (AOIS) in serial multistage manufacturing processes. The MMAS is an improved version of the Ant System, which was proposed by Stützle and Hoos for combinatorial optimisation problems. It was designed to have a relatively long initial exploration phase with a subsequent transition to an intensive exploitation phase. To improve the performance of the MMAS algorithm, the max-min ant system needs to be enhanced with local search. Six local search methods which are well-known and suitable for the AOIS problem are used. The aim of the methods is to create improved inspection plans. Also, two novel heuristics information for the MMAS algorithm have been created. The heuristic information for the MMAS algorithm is exploited as a novel means to guide ants to build reasonably good solutions from the early stages of search of the algorithm.

6.1 Allocation of the inspection stations problem

Figure 6.1 schematically represents the concept of serial processing workstations in a serial multistage manufacturing process. Actual production strategies, similar to this representation, are typical of batch manufactured products. In the general case, there are *n* discrete processing stages through which the work in-progress is routed in a fixed sequence. The first processing workstation receives the raw materials in batches of a certain size, and the last workstation is involved in shipment of the final product. As the items within a batch move through the processing workstations, they may incur defects. The purpose of inspection is to separate conforming product units from non-conforming ones to avoid further processing of

items that are already defective. Each inspection station is characterised by its inspection cost. The objective is to allocate limited inspection stations in a serial multistage manufacturing process. As a result, the total manufacturing cost of a product can be reduced without affecting the quality of the product. The total cost is defined as the sum of the costs of production, inspection, internal and external failure costs. Every processing workstation entails a constant unit processing cost. Each processing workstation has a known probability of resulting in a production error. At each inspection operation, two types of inspection errors may occur with known probabilities: classification and subsequent disposal of a conforming unit as non-conforming, and classification of a non-conforming unit as conforming, allowing it to proceed to the next operation in the system. Units classified as non-conforming are removed from the mainstream flow and are scrapped or reworked. Once an item is identified by an inspection station as being defective, it is assigned to a repair facility at the same processing workstation.

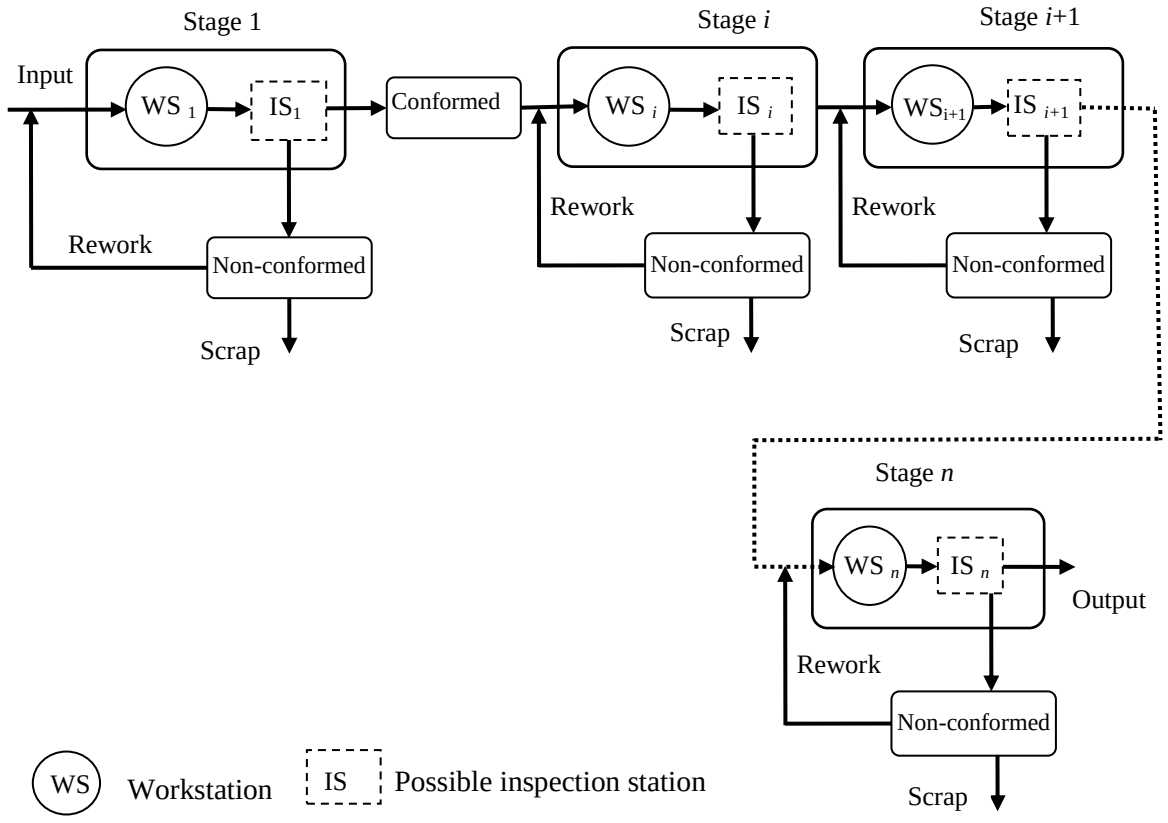


Figure 6.1: Structure of a serial multistage production system with inspection stations

The rework option consists of the execution of a repair operation on the units classified as non-conforming. The repair operation entails the addition of a constant unit repair cost to each unit. After its completion, the repaired units are merged with the units classified as conforming, so that no distinction between the repaired units and the others remains. At the end of the process, the external failure cost represents repair and replacement that is incurred for each non-conforming unit that exits the system.

6.2 Max-min ant system for the inspection allocation problem

The standard MMAS presented by Stützle and Hoos (2000) was derived from the standard ACO. The MMAS algorithm was described in more detail in section 5.3.4 for the interested reader. The following are the elements of MMAS algorithm for solving the allocation of inspection stations problem:

(1) Pheromone trail initialisation.

(2) Heuristic information.

(3) Construction of solutions.

(4) Selection probability.

(5) Pheromone updating.

(6) Pheromone trail limits.

(7) Termination condition.

These elements are described in the following subsections.

6.2.1 Pheromone trail initialisation

In the AOIS problem, the pheromone trail strength (pheromone trail matrix) is initialised as the maximum $\tau_{\max}$ possible trail strength for all edges. This type of trail initialisation is chosen to increase the exploration of solutions during the first iterations of the algorithm. The trail strength then will reduce due to evaporation. After the first iteration of MMAS, the trails

will be bounded to take values within the specified limits as will be described in section (6.2.6). As only the best ant is allowed to update its tour, and only the trails of arcs participating in the best tours are strengthened or maintained at the upper trail level τ_{max}. Hence, arcs that do not receive any reinforcement will continuously lose their trail strength and be selected more rarely by the ants.

6.2.2 Heuristic information

In this section, the importance, inspiration and method of computation of heuristic information are described. Heuristic information is defined in accordance with the characteristics of the problem that is yet to be solved. The aim is to find appropriate heuristic information for the MMAS algorithm to tackle the AOIS problem, as a result of which a near optimal solution can be obtained.

- **Importance of heuristic information**

In the MMAS algorithm artificial ants are the stochastic solution construction procedures, which are biased by artificial pheromones (τ_{ij}) and heuristic information (η_{ij}), as will be described in section 6.2.4. In the initial stages of the search, the pheromones, initially being set to random values, do not guide the artificial ants in a useful way. This leads to the creation of trails of very poor solution quality. The major task of heuristic information is to avoid this by initially biasing ants so they can build rationally good trails from the very first search of the algorithm. As described in chapter 5, the ACO approach was rather unexplored for the AOIS problem at the time that this research started. There is no heuristic information created for the AOIS problem. In addition, heuristic information is apart from the ACO algorithm, as will be shown in equation (6.3). Thus there is a need to create heuristic information for the AOIS problem. The significance of the contribution in this section is that the heuristic information makes ACO algorithms (AS, ACS, Ant-Q and MMAS) more efficient in solving real-world problems in a number of different areas of the AOIS problem. Examples are the

rigor of the inspections (acceptance limits) for each inspection station, the number of inspections executed (sample size-sampling frequency) for each inspection station and these issues are able to include different production configuration such as assembly and non-serial. In addition, heuristic information increases ability of the ACO to find high-quality solutions to AOIS problems in a reasonable time. Angus (2008) indicated that the use of a heuristic value, whenever possible, considerably improves ACO performance. Furthermore, by introducing heuristic information the probable search space (the search space most likely to be explored) becomes much smaller than the original search space.

- **Inspiration of the heuristic**

As described in the literature review in chapter 5, in many cases the heuristic information is the cost, or an estimate of the cost, of adding the component or connection to the solution under construction. Different problems required different heuristic information. In the AOIS problem there are many costs incurred by inspection operations or by processing operations resulting from passing the raw materials through a sequence of processing workstations. These costs are the inspection cost, replacement cost, reworking cost, manufacturing cost, penalty cost and scrap cost. The problem is which of these costs to use and how to link them with the heuristic value. Therefore, attention should be given during the selection of these costs. In an AOIS problem, the defect rate generated at each processing workstation has a great impact on the characteristics of the AOIS problem. This is because the defective items lead to an increase in the total manufacturing cost. This cost is increased considerably if these defects are allowed to pass through subsequent processing workstations. It should be noted that the aim of heuristic information is to guide the ants when assigning inspection stations to workstations, as a result of this the total cost is minimised. Placing inspection stations after workstations that generate highly defective items lead to minimising the total manufacturing costs.

On the other hand, it is impractical to select external failure costs (replacement and repair) as a guide to the heuristic information. This is because the external failure cost only occurs when defective items reach the customer. In addition, external failure cost usually occurs only at the last workstation. In the case of reworking cost, it cannot be taken as a guide to heuristic information toward promising regions of the search space. This is because the reworking cost is usually less than other costs, such as operation or scrap costs. As a result, using the rework cost as heuristic information leads to ineffective inspection positions. In the case of scrap cost, it is dependent on the complexity and value of the product. Also it is ineffective to use scrapping cost as a guide to the heuristic information. On the other hand, operation cost is an important characteristic for the AOIS problem. This is because the operation cost is calculated for all items processed at every processing workstation, regardless of whether or not an inspection is performed at any of the processing workstations. Therefore, placing inspection stations before these workstations will minimise the total cost through the detection of defective items, before wasting additional costs by continuing to process them. Inspection cost is also an important characteristic for the AOIS problem. It is clear that the minimum inspection cost leads to the minimum total cost of a product.

It is evident that operation cost, defect rate and inspection cost are the most appropriate factors to be considered as guides to heuristic information for the best inspection plans of the search space. Heuristic information can be described as a rule of thumb which serves as a guideline for generating a solution for the AOIS problem. Figure 6.2 shows the inspiration of the heuristic information for the AOIS problem. Two novel heuristic methods are created to guide the ant to locate an inspection station to a workstation based on the concerns of operation cost (U_k), inspection cost (IC_k) and defect rate (Z_k), respectively. These methods are the operation cost and defect rate method (OCDM), and the Scores Method (SM).

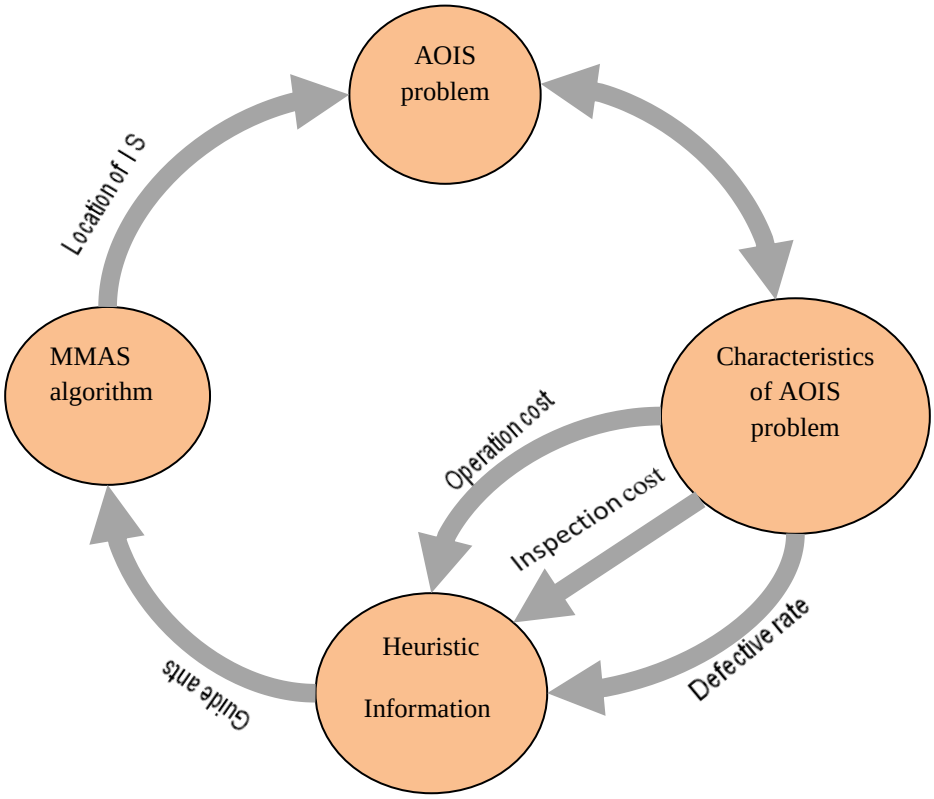


Figure 6.2: Inspiration of heuristic information for AOIS problem
IS: inspection station

The originality of the contribution in this section is that these methods have never been applied to this type of AOIS problem or in different areas. In particular, the SM was first used by Shetwan et al. (2011) as a new heuristic method to solve the AOIS problem as a part of this research. Previous heuristic information methods were based on a simple idea such as in TSP. However, the concept of the two heuristic methods is more complex and very different from those in the previous literature. Further to investigation of the characteristics for the AOIS problem which were described above, a number of steps are required to reach to the development of these two heuristic information methods as will be described in next subsections.

There are several methods to compute the heuristic information of each move. Since the heuristic information is calculated for all moves in all ants it may significantly reduce the

efficiency of the algorithm, and thus should be computed in an efficient manner. In the earlier implementations of ant algorithms, heuristic information was calculated either *a priori* or *a posteriori* (Chaharsooghi, 2008). In the first category of implementations, the static heuristic information is first calculated at the start and remains unchanged during the running of the algorithm. In the second category, the dynamic heuristic information depends upon the current state of the ant. Two contradictory aspects are to be considered in the calculation of heuristic information. These are: (1) the efficiency of calculation, and (2) the quality of information. The implementations with *a priori* heuristic information are efficient but do not thoroughly indicate the desirability of moves. On the other hand, one advantage of using a dynamic heuristic method is that the precise estimation of the desirability of each move is obtained, although the efficiency of computation is not satisfactory. In this research, a novel method has been developed for calculating heuristic information for the AOIS problem by considering these aspects.

The heuristic information should, intuitively, prefer workstations which have a high operation cost, high defect rate and low inspection cost. This avoids processing items that are already defective by continuing to process them, otherwise unnecessarily greater costs will incur. In the AOIS problem, moving the ant from the current node to the next adjacent node does not influence the values of the operation cost, inspection cost or defect rate. Hence the proposed heuristic information is assumed to be a static heuristic value and can be pre-computed before applying the algorithm.

- **OCDM**

The OCDM is based on the operation cost (U_k), inspection cost (IC_k) and defect rate (Z_k). To use these costs and the defect rate in the developed algorithm there is a need to link them via mathematical formula. Clearly, by multiplying the operation cost (U_{k+1}) with the defect rate (Z_k) at each workstation results in different cost components throughout the serial line. For

example if a workstation k has a higher defect rate compared to the other workstations, and workstation k+1 has a higher operation cost compared to the other workstations, this leads to a higher desirability for allocating an inspection station at workstation k. In contrast, the minimum inspection cost at workstation k leads to a higher desirability for allocating an inspection station at workstation k. Therefore, the heuristic information is calculated by considering these characteristics (operation cost, defect rate and inspection cost). The heuristic information pertaining to move $v = (i, k)$ is denoted by η_{ik}. This move indicates the desirability of locating inspection station i to workstation k as shown in equation (6.1).

$$\eta_{i,k} = \frac{[U_{k+1} \times (1 + Z_k)]}{IC_k} \qquad (6.1)$$

If a workstation k has a higher defect rate, low inspection cost and workstation (k+1) has a higher operation cost, this workstation k has a greater probability of being selected.1 is added to equation (6.1) for avoiding the result of 0, in cases where the defect rate is zero.

To apply the OCDM, a multistage manufacturing process is considered, consisting of five workstations arranged in a serial manner using real data and involving manufacturing of pistons. This case study introduced by Kaya and Engin (2007) to define the sample size at attributes control the chart in multistage processes. The piston is one of the most important moving components in the engine. Pistons whose casting stages are completed are processed on machines which are equipped with CNC machines on five different workstations. These operations, such as processing, turning and drilling, are shown in Figure 6.3. The defect rate, unit inspection cost and unit operation cost at each workstation are given in Table 6.1. It should be noted that the drilling machine does not produce any defect.

It is assumed that there are three inspection stations to be located in the serial line. It is also assumed that during the search of the algorithm an ant is randomly placed on the third

workstation (WS3), this ant then has many choices to move to locate the second inspection station, as shown in Figure 6.4.

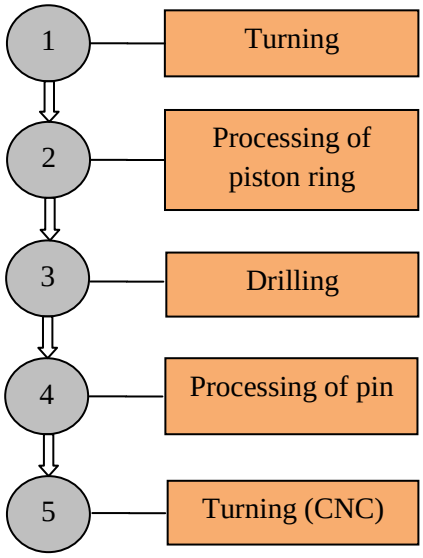


Figure 6.3: Flow of piston production (Kaya and Engin, 2007)

Table 6.1: Unit operation cost, unit inspection cost and defective rate for piston production

	Workstation (WS)				
	1	2	3	4	5
Unit operation cost (U_k)	25	100	150	50	60
Unit inspection cost ($IC_{,k}$)	30	20	25	35	50
Defect rate (Z_k)	0.01	0.014	0.0	0.028	0.071

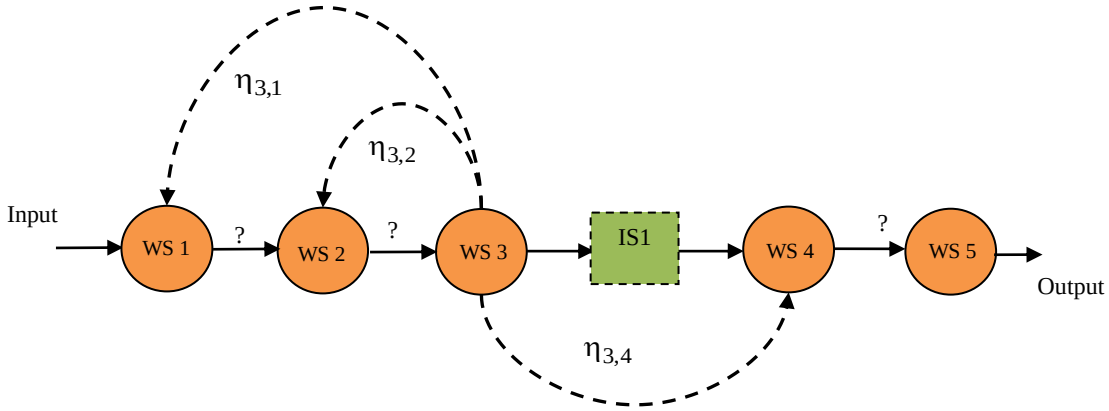


Figure 6.4: The choices for an ant at WS3 to locate the next inspection station

The heuristic information will guide that ant to move to the next workstation that has a higher defect rate, operation cost and low inspection cost. Using equation (6.1) the following results show all possible moves for that ant:

$$\eta_{3,1} = \frac{[100 \times (1 + 0.01)]}{30} = 3.366 \qquad \eta_{3,2} = \frac{[150 \times (1 + 0.014)]}{20} = 7.605$$

$$\eta_{3,4} = \frac{[60 \times (1 + 0.028)]}{35} = 1.762$$

In the MMAS algorithm the heuristic information is superscript to beta $[\eta_{ij}]^{\beta}$ and $\beta \geq 0$ as will be shown in equation (6.3). Based on these calculations the sequence of workstations that have a higher probability of being selected by the ant placed on WS3 is: WS2 and then WS1. These workstations have a higher desirability ($\eta_{3,2} = 7.60, \eta_{3,1} = 3.36$) for the ant. The pseudo-code of the developed OCDM heuristic information is outlined in the following procedure:

Procedure of heuristic information

Initialise*: U_{k+1}, and Z_k and number of workstations*

For *j*=1*: number of workstations*

Calculate heuristic information using equation (6.1)

End

End Procedure

Return heuristic information

- **Scores method (SM)**

The SM is based on the operation cost (U_k) and defect rate (Z_k). The importance of using these rules is respectively to avoid processing defective items in subsequent operations and to avoid the high processing costs for items that are already defective. The question is how to combine these two rules together to guide the ants to locate inspection stations. Consider the same multistage manufacturing process consisting of five workstations arranged in a serial

manner using real data involving the manufacturing of pistons. The unit operation cost (U_k) and defect rates (Z_k) at each workstation are given in Figure 6.5. Assuming that during the algorithm search an ant was placed on the fourth workstation (WS4), based on these two rules the ant has two choices of movement to locate the next inspection station, as shown in Figure 6.5. These choices are whether to allocate an inspection station after WS5, which has a high defect rate (Z_5=0.071), or before WS3, which has a high operation cost (U_3=150). The ant must only move to one workstation to assign an inspection station. This leads to the use of a SM which is a combination of the two choices in one.

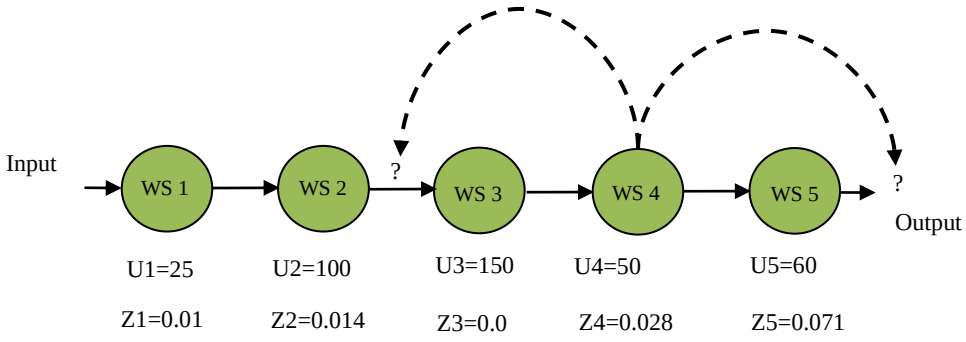


Figure 6.5: Importance of using the Scores Method

The SM is created to allow the heuristic information to guide the ant and locate an inspection station based on the concerns of operation cost (U_k) and defect rate (Z_k). The scores to locate an inspection station before or after workstation *k* are determined as $S_{U,k}$ and $S_{Z,k}$. Those two kinds of score represent how important it is to allocate an inspection station for detecting workstation *k* by individually considering the operation cost and defect rate. The two upper limits of the scores are assumed to be equal to *n* (*n*=number of workstations). After that, these values are decreased gradually by 1 in each assigned score to the workstations based on the higher values of U_k and Z_k. As shown in Figure 6.1, there are *n* workstations in a serial multistage manufacturing system, and either $S_{U,k}$ or $S_{Z,k}$ are determined to be *n*, *n*-1,. . . ,1.

In the case of operation cost, the higher the U_k, the greater the priority to allocate (higher score) an inspection station at the previous workstation *k*-1. The larger the $S_{U,k}$, the higher the priority to locate an inspection station at workstation *k*. An earlier workstation will have a higher priority if two or more workstations have equal operation costs. Considerations should be given that $S_{U,k}$ is always set at the earliest workstation. This avoids the processing of already defective items, otherwise unnecessarily greater costs will incur. Wild (1989) pointed out that the accumulation cost resulting from processing items that are already defective increases as the number of processing workstations increases.

In the case of the defect rate, the higher the Z_k, the larger priority to allocate (higher score) an inspection station at workstation *k*. The larger the $S_{Z,k}$, the higher priority to locate an inspection station after workstation *k*. If two or more workstations have the same defect rate, the higher priority to locate an inspection station is given to the last workstation. This will guarantee the best quality of product sold to the customer. On the other hand, if two or more workstations have the same defect rate and none of them is the last workstation, an earlier workstation will have a higher $S_{Z,k,}$ priority to locate an inspection station after workstation *k*. This avoids further work on scrapped units. After that, the total score ($S_{a.k}$) of locating an inspection station for each workstation will be: $S_{a,k}= S_{U,k}+ S_{Z,k}$. The higher the total score $S_{a,k}$ is for the workstation, the higher the priority for placing an inspection station at it.

Table 6.2 shows an example for determining these scores. As can be seen from Table 6.2 that workstation 3 has a higher operating cost (U_3=150) than the other workstations. This gives priority to assign high score ($S_{U,\ 2}$ =*n*, $S_{U,\ 2}$ =5) at the previous workstation *k*-1 (workstation 2). Also workstation 2 has a higher operating cost (U_2=100) than the other workstations. This also gives priority to assign high score ($S_{U,\ 1}$ =*n*-1, $S_{U,\ 1}$ =4) at the previous workstation *k*-1 (workstation 1). The same procedure is applied to the rest of workstations. In case of defect rate Z_k, workstation 5 has a characteristically high defect rate (Z_5=0.071) than the other

workstations. This gives priority to assign high score ($S_{Z,5}=n$, $S_{Z,5}=5$) after workstation 5. Also workstation 4 has a characteristically high defect rate ($Z_4=0.028$) than the other workstations. This also gives priority to assign high score ($S_{Z,4}=n$-1, $S_{Z,4}=4$) after workstation 4. The same procedure is applied to the rest of workstations. The total scores at each workstation ($S_{a,k}$) will be: $S_{a,1}=S_{U,1}+S_{Z,1}=4+2=6$, $S_{a,2}=S_{U,2}+S_{Z,2}=5+3=8$. The same procedure is applied to the rest of workstations. The final step is to determine the priority (P_k) to the total scores for each workstation. Clearly, the total scores at workstation 2 ($S_{a,2}=8$) is higher than the others. This gives the higher priority to the workstation 2 ($P_2=1$) to allocate an inspection station (the first inspection station) at it. Also the total scores at workstation 4 ($S_{a,4}=7$) is higher than the others. This also gives the higher priority to the workstation 4 ($P_4=2$) to allocate an inspection station (the second inspection station) at it. The same procedure is applied to the rest of workstations.

Table 6.2: Example of determining scores for the heuristic information

	Workstation (WS)				
	1	2	3	4	5
Operation cost (U_k)	25	100	150	50	60
Scores of operation cost ($S_{U,k}$)	4	5	2	3	1
Defect rate (Z_k)	0.01	0.014	0.0	0.028	0.071
Scores of defect rate ($S_{Z,k}$)	2	3	1	4	5
Total score $S_{a,k}=S_{U,k}+S_{Z,k}$	6	8	3	7	6
Priority (P_k)	4	1	5	2	3

$$\eta_{ij}=\frac{1}{1+P_k} \qquad (6.2)$$

Considering the same example of multistage manufacturing process described above for piston production that used real data. The defect rates and unit operation cost at each workstation are given in Table 6.2. The sequence of workstations based on the SM (higher priority) is: WS2, WS4, WS5, WS1, and WS3. In the MMAS algorithm the heuristic information is superscript to beta $[\eta_{ij}]^{\beta}$ and $\beta \geq 0$, as will be shown in equation (6.3). In this

case, if $\eta_{ij} = P_k$ the highest desirability for allocating an inspection station goes to workstation 3, which has priority (P_k=5) (the lowest priority), then workstation 1 which has priority (P_k=4), and so on. To ensure that the heuristic information will guide the ant to the right workstation with the higher priority (P_k=1, 2,..., n) as determined in Table 6.2, the heuristic information is calculated by taking the inverse of P_k as shown in equation (6.2).

In this case, the workstations with higher priority have a greater probability of selection. 1 is added to avoid dividing by 0. To apply the developed heuristic information on the serial multistage manufacturing process which produces pistons and is based on real data, as shown in Figure 6.6, it is assumed that an ant placed randomly on WS4. Subsequently that ant has many choices to assign the next inspection station. Based on the data in Table 6.2 and by using equation (6.2) all possible moves for that ant are computed as follows:

$$\eta_{4,1} = \left[\frac{1}{1+4}\right] = 0.2 \quad \eta_{4,2} = \left[\frac{1}{1+1}\right] = 0.50 \quad \eta_{4,3} = \left[\frac{1}{1+5}\right] = 0.166 \quad \eta_{4,5} = \left[\frac{1}{1+3}\right] = 0.250$$

It is assumed that there are three inspection stations available to be distributed along the line. Then the workstations with highest priority are WS2 ($\eta_{4,2}$) and WS5 ($\eta_{4,5}$). Therefore, the first and the second inspection stations will be placed after WS2 and WS5 respectively, and the third inspection station is already placed randomly after WS4.

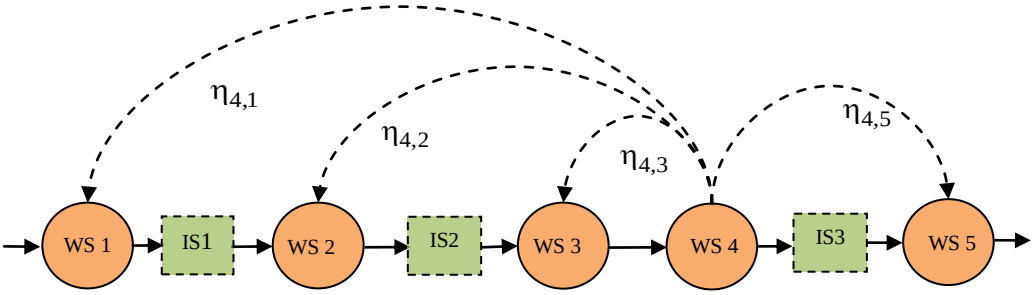


Figure 6.6: All possible moves from WS4 to locate inspection stations based on P_k

The pseudo-code of the developed SM heuristic information is outlined in the following procedure:

Procedure of heuristic information

***Initialise**: U_k, Z_k and number of workstations*

***For** i=1: number of workstations*

Assign scores for U_{k-1}, and Z_k %Higher U_{k-1} or Z_k is the higher priority to assign scores%

End

***For** k=1: number of workstations*

Calculate $S_{a,k} = S_{U,k} + S_{Z,k}$

End

Determine priority P_k

End Procedure

Heuristic information =1/1+P_k

6.2.3 Construction of the solution

A feasible and complete solution of the formulated AOIS problem is considered as a static connected graph G=(C, L) where C is the set of nodes or workstations where [C=1,...*n* (*n*=number of workstations)] and L is the set of undirected arcs connecting them as shown in Figure 6.2. Each arc is weighted by a pair of numbers $\{\tau_{ij}, \eta_{ij}\}$, where τ_{ij} is the pheromone trail level and η_{ij} is the heuristic information as described in the previous section. For the tour construction, initially each ant *k* is placed on a randomly chosen workstation (WS). For example, when an ant is placed on workstation *i* as shown in Figure 6.7, then, starting from that workstation, an ant moves to a still unvisited workstation according to the probability distribution as will be described in the next section until a tour is completed. An ant stochastically prefers to move to workstations which are high heuristic (η_{ij}) values and which are connected by an arc with a high pheromone value. The move of that ant is not affected by the feed-forward of the manufacturing process but it is affected by the arcs

between (workstations or nodes) which have high strength of pheromone trails and high desirability based on heuristic information.

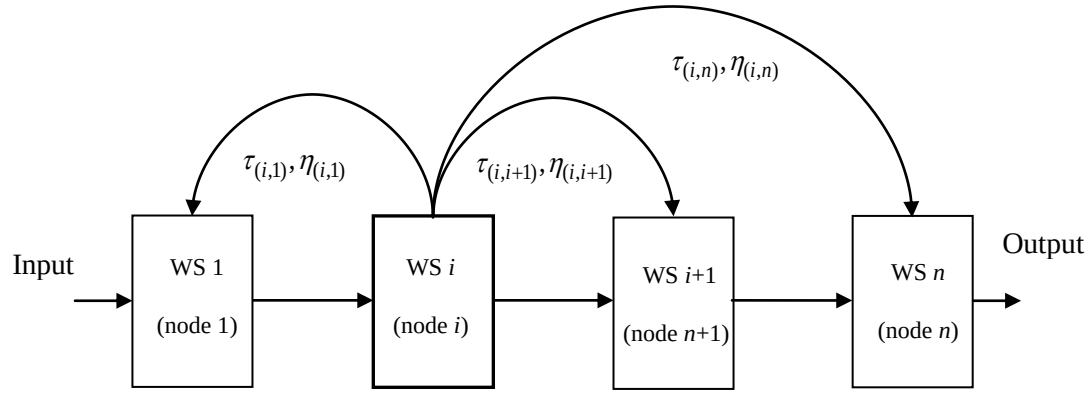


Figure 6.7: Graph representation of the AOIS problem

Consider that three inspection stations have to be placed along a serial multistage manufacturing process consisting of eight processing workstations. Assume that an ant has visited WS3, WS4 and WS6 in any sequence (forward or backward or combination). It means that the inspection stations are located after WS3, WS4 and WS6. In the MMAS, the algorithm, each ant uses is a list to keep track of the workstations it has visited and the partial tour constructed so far is recorded as well. This list is also used to avoid moves to already visited workstations and for the ant to build legal tours. The list is denoted by $Tabu_k$. Solution construction by artificial ants can then be imagined as a walk over a weighted, fully connected graph where the nodes represent the workstations and the arcs are connections between the workstations. In the AOIS problem, the objective is to find the inspection plan that has the lowest total cost. In the proposed algorithm, it is assumed that each ant initially assigns a limited number of inspection stations to corresponding workstations in serial multistage manufacturing processes.

6.2.4 Selection probability

In all the implementations of ant algorithms, an ant chooses a move to go from its current state (workstation), to the next adjacent state (workstation), based on a rational combination of two factors, namely the desirability (heuristic information η_{ij}) of that move and the quantity of pheromone on the edge which is to be traversed τ_{ij}. In the AOIS problem, ants prefer workstations connected by arcs with a high pheromone trail. In the AOIS problem, when located at workstation *i*, ant *k* moves to workstation *j* chosen according to the probability distribution given by equation (6.3):

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{r \in N_i^k} [\tau_{ir}(t)]^{\alpha} \times [\eta_{ir}]^{\beta}} & \text{if } j \in N_i^k \\ 0 & \text{if } j \notin N_i^k \end{cases} \qquad (6.3)$$

where α and β are the parameters that determine the relative dependence on pheromone trail intensity and local information, respectively, and N_i^k is the feasible neighbourhood of ant *k*, that is, the set of workstations which ant *k* has not yet visited.

6.2.5 Pheromone updating rule

In the MMAS, only one single ant is used to update the pheromone trails after each iteration. Consequently, the modified pheromone trail update rule is given by equation (6.4):

$$\tau_{ij}(t+1) = (1-\rho) \times \tau_{ij}(t) + \Delta\tau_{ij}^{best}(t) \qquad (6.4)$$

$\Delta\tau_{ij}^{\text{best}}(t)$ is defined by equation (6.5).

$$\Delta\tau_{ij}^{best}(t) = \begin{cases} \dfrac{Q}{f(s^{best})} & \text{if arc } (i, j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases} \qquad (6.5)$$

where $f(s^{best})$ denotes the solution cost (inspection plan cost) of either the iteration-best (s^{ib}) or the global best solution (s^{gb}) and *Q* is the total amount of pheromone deposited by an ant

on the edges of the path. However, the particular value of Q does not have a significant influence on the final performance of the algorithm (Dorigo et al., 1996; Zhi-He, 2008). ρ Is the pheromone evaporation rate; in order to avoid unlimited accumulation of the trail, the value of ρ should be ($0 < \rho \leq 1$).

In the AOIS problem, a mixed strategy between s^{gb} and s^{ib} is used to update the pheromone trail. This is done to obtain gradually shifting emphasis from the iteration-best to the global-best solution for the pheromone trail update. A transition between a stronger exploration of the search space early in the search to a stronger exploitation of the overall best solution later in the run can then be achieved (Stützle and Hoos, 2000; Wong and See, 2009).

6.2.6 Pheromone trail limits

Stützle and Hoos (2000) proposed the provision of dynamically evolving bounds on pheromone trail intensities such that the pheromone intensity on all paths is always within a specified range. As a result, all paths will have a reasonably good probability of being selected and, thus, a wider exploration of the search space is encouraged. The MMAS uses upper $\tau_{\max}$ and lower $\tau_{\min}$ bounds to ensure that pheromone intensities are set within a given range. By limiting the range of values for the pheromone trail, this influences the pheromone trails such that one can easily avoid large relative differences between the pheromone trails during the employment of the algorithm. In the AOIS problem, after updating the pheromone trail at the end of an iteration, the following operation will be applied to the pheromone trail on both edges and points as shown in equation (6.6):

$$\tau^{new}(t) = \begin{cases} \tau_{\min}, & \tau^{\text{old}}(t) < \tau_{\min} \\ \tau^{\text{old}}(t), & \tau_{\min} \leq \tau^{\text{old}} \leq \tau_{\max}. \\ \tau_{\max}, & \tau^{\text{old}}(t) > \tau_{\max} \end{cases} \qquad (6.6)$$

The upper and lower pheromone bounds are calculated as shown in equations (6.7 and 6.8).

$$\tau_{\max} = \frac{1}{1-\rho}\,\frac{1}{f(s_{opt})} \qquad (6.7)$$

where $\tau_{\max}$ is the maximal pheromone trail and s_{opt} is the estimated optimal inspection plan that has the lowest cost.

$$\tau_{\min} = \frac{\tau_{\max} \times (1-\sqrt[n]{p_{best}})}{avg \times \sqrt[n]{p_{best}}} \qquad (6.8)$$

where $\tau_{\min}$ is the minimal pheromone trail and p_{best} is the probability; the best values for p_{best} are 0.05 (Ridge, 2007 and Stützle and Hoos, 2000). *avg* is the average number of options available the ant has to choose from at any decision point (an ant has to choose among $\frac{n}{2}$ workstations (*n*= number of workstations)). The best solution found is constructed with a probability p_{best} which is significantly higher than 0.

6.2.7 Termination condition

Algorithms require a termination condition to control the computational time, similar to other metaheuristics such as genetic algorithms, Tabu search, simulated annealing. This can be done in a number of ways, e.g. repeating the algorithm for a maximum number of iterations, running for a stipulated time and the maximum CPU time has been spent. In this research, it was decided to run the algorithm until the maximum number of algorithm iterations had been reached. This type of termination was used by many of studies (e.g. Liang and Smith, 2004, Ning, 2010, Thepsonthi and Özel, 2012).

6.3 Improving constructed solutions through local search

As described in chapter 5 that none of the previous metaheuristics methods in the literature review used local search to improve their performance. It has been shown that in other area

metaheuristics with local search perform better than metaheuristics without local search optimisation (Duda, 2006). Particularly, in ACO, it has been shown that local search applied to the solutions that the ants have constructed can improve the performance of an ACO algorithm, and to yield high quality solutions (Merkle et al., 2002). In addition, the local search is part of the *DaemonActions* of the ACO algorithm as described in chapter 5.

The local search mechanism known as iterative improvement is used to improve the performance of the MMAS algorithm. It replaces the current solution with a better one and stops as soon as no improved neighbouring solutions can be found. In the AOIS problem, while running the MMAS algorithm when all inspection stations have been allocated to workstations, a complete solution has been constructed by an ant. The constructed solution is represented in the inspection plan with its total cost. Because of the metaheuristics methods which were used to solve the AOIS problem none of them used local search. Therefore, many local search methods in other problems such as vehicle routing problems, quadratic assignment problems, travelling sales man problems and job scheduling problems are investigated. In this research, six neighbourhood structures which are well-known in these problems are used to improve the performance of the MMAS algorithm (Goksal et al., 2012; Deroussi et al., 2006). Consider a serial multistage manufacturing process consisting of six workstations. Figures 6.3-6.8 show the inspection plan which is constructed by an ant (old inspection plan) for the six workstations where 0: no inspection and 1: an inspection is located after a workstation *k*. The constructed inspection plan is improved by applying six neighbourhood structures. These neighbourhood structures are explained as follows:

Crossover: a two-point crossover sets two crossover points at random, and takes a section between the points from the inspection plan. In the following example, the two crossover points are set after the second and fourth inspection stations respectively. The symbol | indicates a crossover point. The resulting two-point crossover has the effect of dividing the

inspection plan into three parts. Recombining the three parts result in the creation of a new inspection plan, as shown in Figure 6.8.

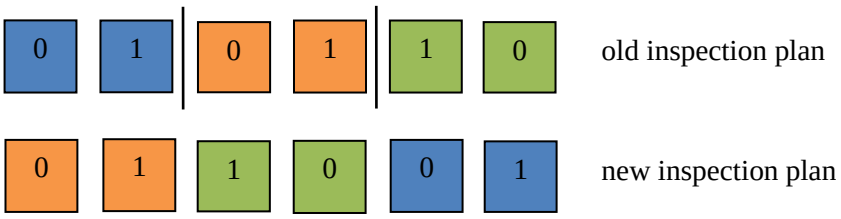


Figure 6.8: Two-point crossover

Interchange: randomly interchange two inspection stations that may not be adjacent in the created inspection plan, as shown in Figure 6.9.

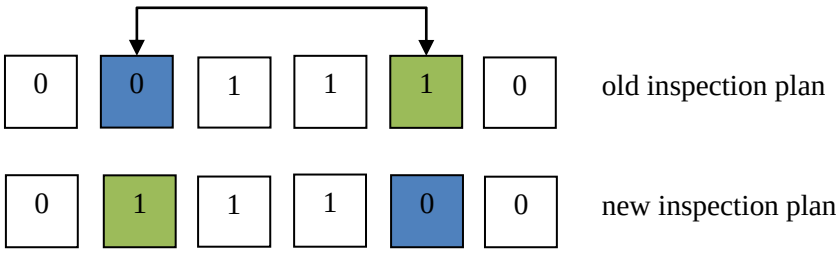


Figure 6.9: Interchange two inspection stations

Swap: two neighbourhood inspection stations are swapped randomly in the created inspection plan, as shown in Figure 6.10.

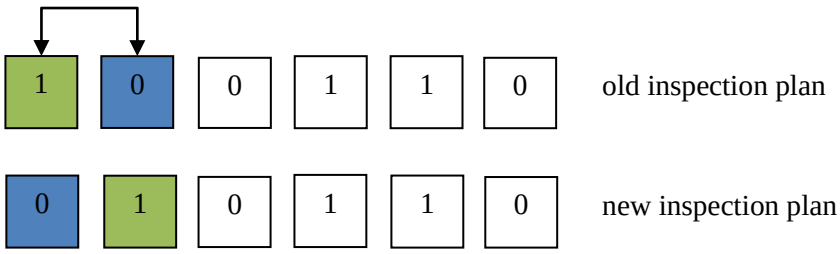


Figure 6.10: Swap two neighbourhood inspection stations

Single insertion: one inspection station is picked randomly and then inserted into all the positions of the created inspection plan, as shown in Figure 6.11.

Delete and add: delete one inspection station randomly in the created inspection plan, and then add it randomly in a new position in the inspection plan, as shown in Figure 6.12.

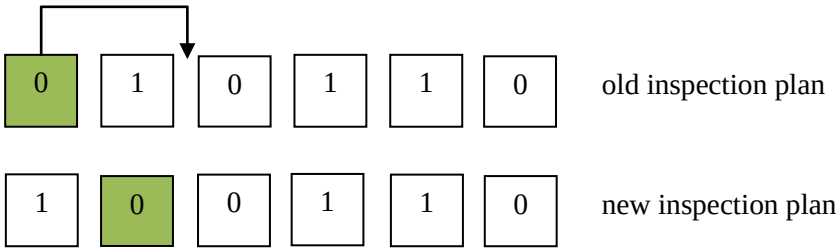


Figure 6.11: Insert one inspection station through the inspection plan

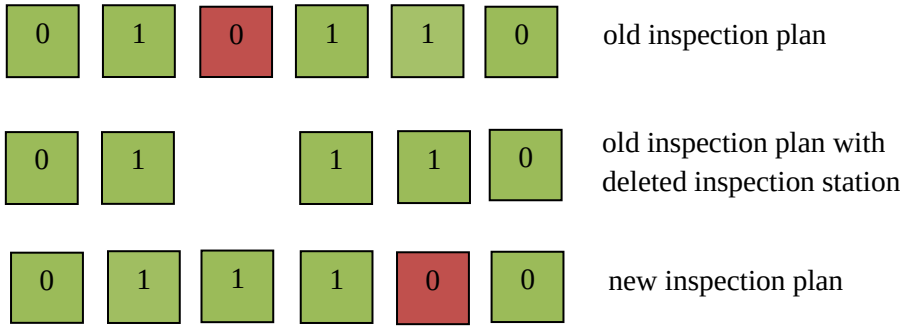


Figure 6.12: Delete and add one inspection station

Block insertion: two inspection stations are picked randomly and then inserted into all the positions of the created inspection plan, as shown in Figure 6.13.

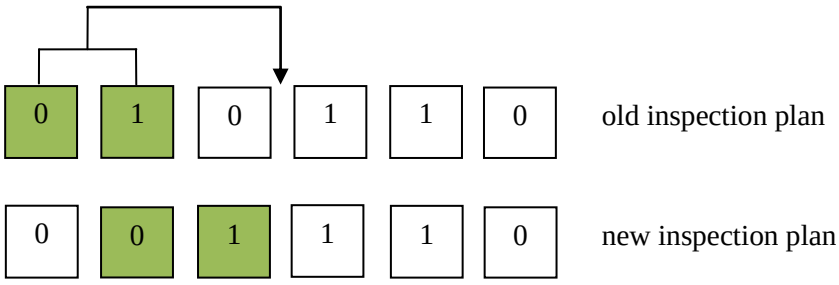


Figure 6.13: Insert two inspection stations through the inspection plan

To yield a further reduction in run-time and to focus the local search around the part where potentially improvement can be found *don't look bits* are used. Don't look bits were first used in the context of local search for the TSP in Bentley (1992). More recently, it was used by Ferreira et al. (2012). This technique is extended in a straightforward way to the AOIS problem. For the AOIS problem, every inspection station is associated with a don't look bit. When first applying local search, all the don't look bits are turned off (set to 0). If for an inspection station no improving move is found, the don't look bit is turned on (set to 1) and the inspection station is not considered as a starting inspection station in the next local search iteration. If an inspection station is involved in a move and changes its location, the don't look bit is turned off. It should be noted that by applying the local search method to the created inspection plan (old inspection plan), the total cost of the new inspection plan is changed. This research focuses on minimising the total manufacturing cost of a product. The objective function (total cost) is calculated for each new inspection plan created by applying local search. Whenever an improvement of the objective function is detected, the new solution replaces the old one, and the process continues until no further improvement is seen. The improved solutions are then used to update the pheromone trails. All these neighbourhood structures are applied to the MMAS algorithm and then compared in terms of the solution quality and processing time needed by each of these local searches.

There exist a large number of possible choices when combining local search with ACO algorithms. Some of these possibilities relate to the fundamental question of how effective and how efficient the local search should be. In fact, in most local search procedures, the better the solution quality returned, the higher the computation time required. This translates into the question whether for a given computation time it is better to frequently apply a quick local search algorithm that only slightly improves the solution quality of the initial solutions,

or whether a slow but more effective local search should be used less frequently. These issues will be considered in the conducted experiments in chapter 8.

6.4 Pseudo-code of the max-min ant system

In this section, the elements discussed above are synthesised to evolve the proposed MMAS algorithm. The pseudo-code of the developed max-min ant system for the AOIS problem is outlined in the following procedure:

Initialise

Set values of parameters number of ants, max-iterations, α, ß and ρ

Initialise pheromone trails matrix

Calculate the heuristic information

While** (termination condition not met) **do

Randomly place each ant in one node

***For** k = 1 to A (number of ants)*

***For** i =1 to number of inspection stations*

Choose the next workstation to visit according to the stochastic decision rule using equation (6.3)

Assign inspection station

End for

End for

Calculate the objective function (total cost) S

Apply local search S'

If $f(S') < f(S)$

$S = S'$

End

Update pheromone trails

Check upper and lower pheromone trails

Check stopping criterion

***End While**;*

***Return** the best solution found*

6.5 Conclusion

The MMAS algorithm was developed for tackling the AOIS problem in serial multistage manufacturing processes. The algorithm assumes that there are limited inspection stations available and allocating inspection stations reduces the total manufacturing cost. The MMAS uses a mixed strategy between the best global ant and best local ant to update the pheromone trails in order to obtain the best performance. The advantage of using a constructive algorithm such as the developed MMAS algorithm is that by generating reasonably good initial solutions, the following local search needs fewer iterations to reach a local optimum. Tow heuristic information methods have been created for the AOIS problem. These heuristic information methods are inspired by the characteristics of the AOIS problem, in particular operation cost, defects rate and inspection cost. Six local search methods for the AOIS problem have also been constructed. The MMAS algorithm is combined with local search in order to enhance the performance and to yield a high quality solution. Well tuned parameters are needed to efficiently initiate the MMAS algorithm and a method to find optimal parameter settings has been investigated in the course of this research.

Chapter 7
Case studies selection

The max-min ant system algorithm was developed to tackle the AOIS problem in the previous chapter. To test the effectiveness of the MMAS algorithm, it must be compared against other search methods. An important issue for the comparison of algorithms is the identification of appropriate case studies. In this chapter, different case studies were investigated. The aim was to select appropriate case studies to assess the performance of the MMAS algorithm. A total of 44 cases were taken from the literature and studied in terms of their characteristics. This led to dividing the cases into two groups. The first group includes cases which relatively match with the developed general cost model (GCM) in terms of the assumptions of the developed GCM. As a result, the cases which are more appropriate to the GCM will be selected. The second group includes rejected cases which do not match with the GCM developed, in terms of the assumptions of the developed GCM. The following section describes the characteristics of these case studies followed by the assumptions of the developed GCM.

7.1 Characteristics of the case studies

The aim was to find case studies with characteristics considered appropriate for the GCM. The GCM was developed under the assumption of a limited number of inspection stations available. Similarly, the objective of almost all the case studies reviewed was to determine inspection points in a serial multistage manufacturing process in order to minimise the total cost. The characteristics of the case studies include production configuration, number of workstations, number of inspection stations, manufacturing cost, inspection errors, inspection

cost, internal failure cost, external failure cost, constraints and solution approach. All these characteristics were described in details in chapters 2 and 3.

Most of the characteristics which were included in these case studies are described above. The aim was to find suitable case studies to assess the developed MMAS algorithm. The assessment of the case studies will be based on the assumptions of the GCM.

7.2 Assumptions of the general cost model

The developed GCM is focused on a serial multistage manufacturing process. It takes into account the constraints on inspection resources. As a result, the total manufacturing cost of a product can be reduced without affecting the quality of the product. The GCM for a serial multistage manufacturing process is based on the following assumptions which should be taken into consideration during the assessment of the case studies:

- Production configuration is assumed to be a serial multistage manufacturing process.
- There is a limited number of inspection stations (e.g. a limited budget) to be distributed in the production line.
- The aim of the objective function is to minimise the total manufacturing cost while maintaining the quality of the product. The total system cost is the sum of the total cost of processing and inspecting the parts produced in the system.
- Sample inspection is used if an inspection station is located after a workstation in the sequence.
- No more than one inspection station can be assigned after each workstation.

Table 7.1 shows the case studies which are relatively closely well-matched to the GCM. Each case study is represented by the first author's name followed by a two-digit publication year. The order of case studies is organised in chronological order by the year in which the paper is published. It should be noted that (Yes) means that this characteristic was included in the case

study and (-) means that this characteristic was not included in the case study. To simplify the selection procedure, the following criteria were used to select the appropriate case studies:

- **Data availability**

As can be seen from Table 7.1, there are some data which were not considered in the developed models used by the authors, such as replacement cost and external cost in case study (1). This might have occurred for the sake of simplicity in order to allow a tractable formulation model and solution. On the other hand, some data in the case studies are missing. For example, in case study (3), some data were not given, such as the quantity of items entering the system, the number of inspection stations and the penalty cost. In case study (5), the authors included the external failure cost in their model (replacement and repair costs), but values were not given for the experiment parameters or the scrap cost in the internal failure cost. In case (6), the unit inspection cost was not provided by the authors. In case study (12), the case included only inspection and scrap costs but did not provide the input material (number of items entering the system), unit scrap cost or unit inspection cost.

- **Matching to the general cost model**

In this section, the purpose is to look for cases that considered similar assumptions as the developed GCM. In addition, how these assumptions were treated through the case study was evaluated, taking into consideration that some assumptions might be adapted or reconsidered. For example, some case studies assumed that inspection was error free. In the GCM in chapter 3, both types of errors were incorporated into the GCM. This assumption was adopted in the developed GCM simply by setting parameters of inspection errors $\alpha=0$ and $\beta=1$. In case study (1), the authors were interested in the reliability of each machine and the inspection time. These issues were not included in the developed GCM. In case study (2), although the case considered a limited number of inspection stations, they assumed that multiple inspection stations might be located after a workstation.

Table 7.1: Characteristics of case studies relatively closely match with the GCM

Case study number	Author	Production Configuration	Number of WS	Number of IS	Man cost	Inspection error	Inspection cost		Internal failure cost		External failure cost		Constraints	Solution approach
							Fixed	Variable	Repair/ Rework cost	Scrap cost	Repair cost	Repl. cost		
1	Park (88)	Serial	4	_	Yes	Free of error	Yes	Yes	Yes	Yes	_	_	_	IP
2	Raz (91)	Serial	10	Multiple	Yes	I and II	_	Yes	Yes	_	Penalty cost		_	B&B
3	Viswanad-ham (96)	Serial and assembly	5-25	_	Yes	I and II	_	Yes	Yes	Yes	Penalty cost		_	GA and SA
4	Bai (96)	Serial	4-10	1-3	_	I and II	_	Yes	Yes	_	Penalty cost		Limited inspection stations& rate of inspection	DP
5	Lee (98)	Serial	_	_	Yes	I and II	Yes	Yes	Yes	Yes	Yes	Yes	Limited inspection stations & inspection time	Heuristics
6	Shiau (02)	Serial	7	2,3,5	Yes	I and II	_	Yes	Yes	Yes	Yes	Yes	Limited inspection stations	Heuristics
7	Langner (02)	Serial	6	_	Yes	I and II	_	Yes	Yes	_	_	_	AOQL	GA
8	Shiau (03a)	Serial	7	3	Yes	I and II	_	Yes	Yes	Yes	_	_	Limited inspection stations	Heuristics
9	Shiau (03b)	Serial	5	3	Yes	I and II	_	Yes	Yes	_	Yes	Yes	Limited inspection stations	Heuristic
10	Rau (05)	Serial	6-16	_	Yes	I and II	Yes	Yes	Yes	Yes	Penalty cost		_	Heuristic
11	Shiau (07)	Serial	5	3	Yes	I and II	_	Yes	Yes	Yes	Yes	Yes	Limited inspection stations	GA
12	Sadegheih (07)	Serial	10	3	–	Free of error	_	Yes	_	Yes	_	_	–	GA
13	Van Volsem (07)	Serial	6	_	_	Free of error	_	Yes	Yes	_	Penalty cost		_	GA

Man.: manufacturing, Repl.: replacement, I and II: Type I and type II inspection errors, AOQL: average of outgoing quality level.

On the other hand, in the GCM, it is assumed that no more than one inspection station can be assigned after each workstation. In case study (4), the case included assumptions such as the rate of processing, the rate of production and the rate of inspection. All these assumptions are not incorporated in the GCM. In case study (7), the objective function was constrained by the average outgoing quality level which is not included in the GCM. In addition, the number of items to be inspected was randomly based on a sample size selected at each workstation. However, in the GCM, when inspection is performed, 100% of items are inspected. In case study (13) the case was interested in determining the rigor of the inspections (acceptance limits) for each inspection station. These issues are not included in the GCM.

The case studies used various optimisation methods to solve the AOIS problem. These optimisation methods included exact and approximate methods. It should be noted that most of these case studies used the total manufacturing cost and processing time to test the performance of their optimisation methods against the optimal solution. The optimal solution was obtained using the complete enumeration method (CEM). The CEM checks for all possible combinations of inspection plans in the search space. As a result, the quality of the solution obtained by a method can be measured by its closeness to the optimal solution. The running time is the processing time required to execute the computer programmes for problem-solving in a computer system. It was found that case studies (8) and (9) were nearly identical in terms of the experiment parameters used, and both of them used the heuristic method to approach the AOIS problem. Because case study (9) is more appropriate for the GCM, and also because case study (8) did not consider the external cost, case study (8) was therefore excluded from consideration.

7.3 Cases not matching the general cost model

Table 7.2 shows the cases which were reviewed and investigated but rejected because their characteristics, constraints and assumptions did not match with the developed GCM.

Table7.2: Characteristics of case studies not match with the GCM

Case study number	Author	Production Configuration	Number of WS	Number of IS	Man cost	Inspection error	Inspection cost		Internal failure cost		External failure cost		Constraints	Solution approach
							Fixed	Variable	Repair/ Rework cost	Scrap cost	Repair cost	Repl. cost		
14	Lindsay (64)	Serial	9	_	_	Free of error	_	Yes	_	Yes	_	_	AOQL	DP
15	White (65)	Serial	6	_	_	Free of error	_	Yes	_	Yes	_	_	_	DP
16	Pruzan (67)	Serial	5	_	Yes	Free of error	Yes	Yes	_	Yes	_	_	_	DP
17	White (69)	Serial	5	_	Yes	Free of error	Yes	Yes	Yes	Yes	Yes	_	_	DP
18	Eppen (74)	Serial	9	_	_	I and II	_	Yes	_	Yes	_		_	DP
19	Enrick (75)	Serial	_	_	Yes	I and II	_	Yes	Yes	_	_		_	DP
20	Ballou (82)	Serial	3	_	_	I and II	_	Yes	_	Yes	Penalty cost		–	NLP
21	Peters (84)	Serial	13	_	Yes	Free of error	Yes	Yes	Yes	Yes	Penalty cost		–	DP
22	Hsu (84)	Serial	4	_	Yes	Free of error	_	Yes	_	Yes	_		_	DP
23	Gunter (85)	Assembly	9	_	–	Free of error	Yes	Yes	Yes	_	_		_	DP
24	Ballou (85)	Serial	3	_	Yes	I and II	_	Yes	_	_	Penalty cost		_	NLP
25	Raz (87)	Assembly	5	_	_	I and II	_	Yes	_	Yes	_	_	I and II	B&B
26	Yum (87)	Serial	3,10	_	Yes	I and II	_	Yes	Yes	Yes	_	_	_	LP
27	Tayi (88)	Serial	5	_	Yes	Free of error	_	Yes	Yes	Yes	Penalty cost		_	NLP
28	Barad (90)	Serial	8	_	Yes	Free of error	_	Yes	Yes	_	_	_	_	Heuristic
29	Raghava-chari (91)	Serial	5	_	Yes	Free of error	Yes	Yes	Yes		Penalty cost		–	DP
30	Chengalur (92)	Serial	3	_	Yes	I and II	_	Yes	_	Yes	Penalty cost		Limited inspection stations	DP

Table7.2: Characteristics of case studies not match with the GCM (continued)

Case study number	Author	Production Configuration	Number of WS	Number of IS	Man cost	Inspection error	Inspection cost		Internal failure cost		External failure cost		Constraints	Solution approach
							Fixed	Variable	Repair/ Rework cost	Scrap cost	Repair cost	Repl. cost		
31	Taneja (94)	Non-serial/serial	5	_	Yes	I and II	_	Yes	_	Yes	_	_	AOQL and Limited inspection stations	GA
32	Jewkes (95)	Serial	_	_	_	Free of error	_	Yes	Yes	_	Yes	_	_	NLP
33	Narahari (96)	Non-serial	4	_	_	Free of error	_	_	Yes	Yes	_		_	NLP
34	Deliman (96)	Serial	5		Yes	II	_	Yes	Yes	Yes	Penalty cost		_	MDP
35	Chen (99)	Assembly	_	_	Yes	Free of error	_	Yes	Yes	Yes	_		_	SA
36	Jang (02)	Serial	6	_	Yes	Free of error	Yes	_	Yes	_	_	_	Limited inspection stations	MDP
37	Kogan (02)	Serial	6	5	_	Free of error	_	Yes	_	_	Penalty cost		_	NLP
38	Emmons (02)	Non-serial	9	_	Yes	Free of error	_	Yes	_	_	_	_	_	Heuristic
39	Hadjinicola (03)	Assembly	3	_	Yes	Free of error	_	Yes	Yes	Yes	_	_	_	NLP
40	Kakade (04)	Serial	2	_	_	Free of error	_	Yes	Yes	_	Penalty cost		Rate of inspection	SA
41	Valenzuela (04)	Serial	2	_	Yes	Free of error	_	Yes	Yes	_	Penalty cost		_	TS
42	Rau et al.(05)	Non-serial	4,8,16	_	_	I and II	Yes	Yes	Yes	Yes	Penalty cost		Limited inspection stations	Heuristic
43	Penn (06)	Assembly	8	_	Yes	Free of error	Yes	Yes	_	Yes	Penalty cost		_	DP
44	Galante (07)	Serial	10	_	Yes	I and II	_	Yes	_	_	Penalty cost		_	GA

For example, some of them studied different system structures such as assembly and non-serial systems, as in case studies (23) and (25). Some others were interested in studying different issues in their case studies. For example, in case study (27), the case study was interested in quality control procedures, in-process inventory and reprocessing. Furthermore, in case study (44), both inspection allocation and operation scheduling were included concurrently. These issues are not included in the GCM.

7.4 Conclusion

Based on these assessments explained above, cases (10) and (11) were selected to be used as testing problems for the developed GCM. Most of the assumptions of the developed GCM were matched by the selected cases. It should be noted that some assumptions will need to be adapted. For example the selected case study (10) used the average deviation from the optimal solution (DFOS) and the standard deviation of DFOS to measure the solution quality of their method. However, the case study did not specify the number of cases generated by each size of workstation. Therefore, in experiments, each size of workstation should randomly generate a number of cases using a uniform random number generator in order to calculate the average DFOS and standard deviation of DFOS. Also, in case study (10), items of external failure cost were represented as aggregated (penalty cost). This assumption will be adapted in the developed GCM. In addition, in case study (10), the number of inspection stations to be located is missing. To adopt this assumption, the developed GCM will be tested with various numbers of inspection stations.

In case study (11), the defective rate and the inspection errors were not specified by the author; therefore, they will be assumed based on similar values used in the earlier literature review in chapter 2. Also, the penalty cost will be assumed to match the complexity of the problem. However, the penalty cost usually depends on the kind of the product produced by the company and its complexity.

All the selected cases tested their optimisation methods by using processing time and total cost against the optimal solution obtained by CEM. The aim of all the selected cases was to minimise the total manufacturing cost. In addition, all the selected cases used inspection of 100% of items if inspection was needed after a workstation. The selected cases will be used to assess the developed MMAS algorithm as presented in chapter 10. To solve the AOIS problem using the developed MMAS algorithm, must first the MMAS parameter values be tuned very well. A method has been proposed to find the optimal combination of the most influential parameter values of the MMAS algorithm. This method is presented in the next chapter.

Chapter 8
Tuning MMAS parameters

The AOIS problem has been approached using a number of different techniques reported in next chapter that vary in their methods and efficiency. These methods are SA, GA, PSO and MMAS algorithm. It is well known that metaheuristic methods are highly dependent on their parameters setting. The metaheuristic is instantiated with several parameters which have to be set manually. Solving problems with the metaheuristic methods is related to the parameter settings concerned in the algorithm, and a good parameter combination will provide the algorithm with overall search capability and a fast convergence rate. In contrast, with unsuitable parameter values, the algorithm convergence will be too fast or too slow, and may prevent the algorithm from finding the best solution. To solve the AOIS problem using the developed these algorithms, parameter values for these algorithms must first be tuned very well. A method has been developed to find the optimal combination of the most influential parameter values of the SA, GA, PSO and MMAS algorithms for tackling the AOIS problem in this chapter.

8.1 Experimental framework

A multistage manufacturing system model consisting of 12 processing workstations arranged in a serial manner was used to allocate five inspection stations. The sample size s_k is set at 80 and the acceptance number (a_k) for workstation k is set at 1. This case study will be labelled as the ‘‘general cost model case study’’. For this number of workstations, the full enumeration of the search space can be generated in a reasonable time. It should be noted that in the AOIS problem, an inspection station is placed only after the workstation which is in need of inspection. For example, assume that the constructed inspection plan for the general

cost model case study was (011110000001); this means that inspection is performed only after the second, third, fourth, fifth and last workstations. Table 8.1 shows the experimental parameters of the developed general cost model and their ranges.

Table 8.1: Experimental parameters for the general cost model case study

Parameters	Range	Brief description
B	1000	Batch size
U_k	[50, 100]	Unit manufacturing cost (£)
IC_m	[40, 50]	Unit inspection cost (£)
Z_k	[0.09, 0.18]	Defective rate
α_m	[0.01, 0.03]	Type-I inspection error
β_m	[0.01, 0.03]	Type-II inspection error
u_k	[30, 120]	Unit scrapping cost (£)
g_k	[40, 80]	Unit reworking cost (£)
δ_k	[0.05, 0.09]	Repairing probability

The experimental parameters were set based on similar experiments in the literature, and the results can be easily interpreted using simple mathematics. These experimental parameters were used to generate 50 different cases. These cases were randomly generated in order to represent the characteristics of different manufacturing systems. In other words, these cases represent 50 different problems. The experimental parameters for the general cost model case study in Table 8.1 are described below:

1. Batch size (B). This is the number of units entering the system.
2. Manufacturing cost (U_k). This is the unit cost of producing one unit at processing workstation k.
3. Inspection cost (IC_m). This is the unit cost of inspecting one unit at inspection station m located after processing workstation k.

4. Probability of non-conforming (Z_k). This is the probability of producing a non-conforming unit at processing workstation *k*.
5. Probability of type I error (α_m). This is the probability that an inspection station *m* placed after processing workstation *k* classifies a CU unit as an NCU, and rejects it.
6. Probability of type II error (β_m). This is the probability that an inspection station *m* placed after processing workstation *k* classifies an NCU as a CU and forwards it for further processing.
7. Unit scrapping cost (u_k). This is the unit cost of scrapping one unit at processing workstation *k*.
8. Unit rework cost (g_k). This is the unit cost of reworking one unit at processing workstation *k*.
9. Probability of repairing a defective unit (δ_k). This is the probability of repairing a defective unit at processing workstation *k*.

8.2 Experimental design

To evaluate the performance of the MMAS algorithm, the results obtained by these methods were compared against the optimal results which were obtained by the complete enumeration method. The aim is to measure deviation from optimal solution (DFOS) for the results obtained by the developed algorithms. In AOIS problem, the total cost (TC) is the sum of processing and inspecting the parts produced in the system as described in chapter 3. Solution quality can be measured by the closeness of the developed algorithms solution to the optimal solution. In other words, the solution quality of a method is indicated by the DFOS. The DFOS is calculated as given by equation (8.1):

$$\text{DFOS} = \left(\frac{\text{TC of algorithm}}{\text{TC of CEM}} - 1 \right) \% \qquad (8.1)$$

The complete enumeration method (CEM) is very well-known and has been used as a benchmark by many of studies in the literature. This is because the CEM checks for all the possible combinations of inspection plans in the search space. An advantage of using the complete enumeration method is that the optimal solution is known. It should be noted that in the AOIS problem, an inspection station is placed only after the workstation which is in need of inspection.

The following steps describe the complete enumeration method:

Step1: Generate all possible location combinations of inspection plans for the general cost model case study. Store the generated inspection plans in the database.

Step2: Filter inspection plans by checking all inspection plans as to whether they fit the constraint to the required number of inspection stations (a limited number of inspection stations). If an inspection plan matches the number of inspection stations required, this inspection plan will be considered. If this condition is not met, the inspection plan will be rejected, because it is unnecessary to check the inspection plans for unsatisfactory assignment location combination. Store the filtered inspection plans in database.

Step3: Evaluate all filtered inspection plans which match the number of inspection stations required. Store the evaluated inspection plans in the database.

Step4: Determine the feasible inspection allocation plan that has the lowest total cost.

CEM was used as benchmark to test the performance of the developed method under the tested parameters. To do so, the MMAS approached the AOIS problem using the specified parameters. The experimental parameters in Table 8.1 were used to generate 50 different cases. These cases were randomly generated in order to represent the varying characteristics of different manufacturing systems. The parameter combination yielded 252 different parameter combinations and the MMAS algorithm was then applied on 50 different cases.

For each of the 50 test cases, 800 evaluations were performed by the algorithms and repeated 30 times to generate average deviation from optimal solution (DFOS) for the test cases.

8.3 Tuning MMAS parameters

The study of the impact of various parameters on the behaviour of ACO algorithms has been an important subject since the first articles by Dorigo et al. (1991a, 1996). Using good parameter values for the MMAS algorithm is important both with the respect to efficiency and effectiveness. Tuning the parameters for any optimization algorithm is at least as important as designing the algorithm itself.

8.3.1 Summary of MMAS parameters

This section will introduce a summary of MMAS parameters.

α: Controls the relative importance of pheromone level (τ_{ij}) of the ant in the process of movement when guiding the ant colony search. Its size reflects the strength of random factors in the path search of the ant colony, and it should be a positive value. A value of zero would turn the algorithm into a standard greedy one, since no pheromone information would be used (Gaertner, 2004). On the other hand, higher values cause the ants to perform too little exploration.

β: Reflects the degree of importance of the heuristic information (η_{ij}) and the importance of the location of inspection stations relative to workstations. With higher β values, a lot of exploration will occur during the execution of the algorithm. In other words, the greater the β value, the greater the chance that the ant will choose the nearest node, and the convergence rate in the search will accelerate, but the algorithm may be easily trapped into a local optima. On the other hand, a β value of zero means that the desirability of locating an inspection station is completely irrelevant, and only the pheromone trail level is used by the algorithm to choose which path to take.

ρ: Controls the evaporation of pheromone in the environment. It indicates the pheromone volatilisation factor, and its size is directly connected to the global search ability and the rate of convergence of the ant colony algorithm. It is intuitively limited by $0 < \rho \leq 1$. Pheromone evaporation reduces the influence of the pheromones deposited in the early stages of the search, when artificial ants can build poor-quality solutions (Dorigo and Stützle, 2004). 1-ρ is the pheromone residue factor, reflecting the degree of interaction between individual ants. When $\rho = 1$, the pheromone is wiped after every iteration. A value of 0 would lead to continually increasing pheromone levels since the pheromone would never evaporate (Gaertner, 2004).

τ_0: Specifies the initial pheromone level on all edges. Intuitively, the pheromone trail strength (pheromone trail matrix) in the MMAS algorithm is initialised as the maximum $\tau_{\max}$ possible trail strength for all edges. This type of trail initialisation is chosen to increase the exploration of solutions during the first iterations of the algorithm. Also, the initial pheromone trail τ_0 could be set to any value, because after the first iteration, the pheromone trail will be forced to fall within $[\tau_{\min}, \tau_{\max}]$.

τ_{max}: This is the maximum possible pheromone trail or the largest possible amount of pheromone added after any iteration. It is calculated by equation (6.7). In the AOIS problem, s_{opt} is the optimal inspection plan that has the lowest cost and is used to determine an appropriate value for $\tau_{\max}$. Clearly, the optimal solution value is not known before running the algorithm and s_{opt} is used as an estimate of that value. Consequently, the upper trail limit is adapted during the running of the algorithm.

τ_{min}: This is the minimum possible pheromone trail, and is calculated by equation (6.8). Many other authors use a different equation to determine $\tau_{\min}$; for example, Ridge (2007)

calculated the lower trail limit $\tau_{\min} = \frac{\tau_{\max}}{2n}$, where *n* is the problem size (e.g. the number of workstations), and Stützle (1998b) set the lower pheromone trail limit to $\tau_{\min} = \frac{\tau_{\max}}{3}$.

***p*_{best}**: This is the probability that an ant will construct the best solution. The experimental results in different areas confirmed that the best value of p_{best} is 0.05. Examples are Afshar (2009) in network optimisation problems, Stützle and Hoos (2010) in QAP and Shuang et al. (2011) in TSP.

numAnts: Represents the number of ants used by the algorithm. Clearly, the more ants used, the slower the algorithm will be. On the other hand, using too few ants means that no meaningful pheromone matrix is created and the search will take more iterations to find the best solution. It should be noted that, when the MMAS was applied to the TSP without considering local search, the best results were obtained by increasing the number of ants proportionally to the instance size. However, this option may not be the best when local search is added to the MMAS algorithm (Stützle, 1998b, Stützle and Hoos 2010).

Q: This is the total amount of pheromone released on all paths by the ants in one cycle. The experimental results conducted by Dorigo et al. (1996) and Zhi-He (2008) showed that the pheromone strength *Q* has no obvious influence on the performance of the ant colony algorithm.

Max-cycles: Controls the number of iterations of the algorithm. Fewer iterations will not be enough to produce good solutions, and increased iterations may lead to unsatisfactory efficiency of the computation. In the AOIS, the MMAS is terminated when the maximum number of algorithm iterations has been reached.

In summary, using good parameter values for the MMAS algorithm is important both with respect to efficiency and effectiveness. With so many parameters, many of which seem to interact non-linearly with each other, establishing the optimal values for all parameters is a

very time-consuming task. From the description of these parameters, it can be seen that the selection of α, β and ρ in the MMAS algorithm has the greatest influence on algorithm performance. Further, many researchers in different area pointed out that these parameters are the most important parameters and have great influence on the performance of MMAS, Ning et al (2010) in layout problems, Fidanova et al. (2011) in Multiple knapsack problems, Wu et al (2011) in QAP, and Liang et al. (2012) in scheduling problems. A method to optimise the most influential parameters α, β and ρ is presented in this chapter.

8.3.2 MMAS parameter settings

The parameter values used in ACO algorithms are often very important in obtaining good results. Dorigo and Stützle (2002) suggested that the exact values of the parameters are often problem-dependent. Unless indicated otherwise, the following parameter settings were used:

α - Controls the relative importance of pheromone. The values: low [0.6, 0.8, 0.9], medium [1, 2, 3, 4, 5] and high [6, 7, 8, 9, 10] of this parameter were tested for tuning the algorithm.

β - Controls the ratio between the importance of pheromone and the importance of the location of inspection stations. The values [0, 1, 2, 3, 4, 5] of this parameter were tested for tuning the algorithm.

τ_0 - Specifies the initial pheromone level on all edges. This matrix of initial pheromone was set to $\tau_{\max}$. Stützle and Hoos (2000) have explained that this should be fairly high to encourage initial exploration during the first iterations.

p_{best} - Probability =0.05 as suggested by Stützle and Hoos (2000) and Ridge (2007).

ρ - Controls the evaporation of pheromone in the environment, and is always set in the range $0 < \rho \leq 1$. ρ was tested with the values [0.01, 0.02, 0.05].

Q- *Q*=*n*, where *n* is the number of workstations (Spiliopoulos and Sofianopoulou, 2008).

max-cycles Controls the number of iterations of the algorithm, the maximum iterations is set at 800.

To explore better solution components and to avoid premature convergence of the MMAS algorithm, a mixed strategy was used to update pheromone trails. In the first 300 iterations, the iteration best ant s^{ib} is used to update the pheromone trails, and then every tenth iteration, the global best ant s^{gb} is used for the pheromone trails update. By gradually shifting the emphasis from the iteration best to the global best solution for the pheromone trail update, a transition between exploration of the search space in early stages to exploitation of the overall best solution in later stages can be achieved.

In the conducted experiments, two possible settings for the number of ants were investigated. In first variant, denoted as $MMAS_{10+ls}$, 10 ants were used and every ant applied local search to its tour. In the second variant, denoted as $MMAS_{+ls+ib}$, the algorithm started with a fixed number of 10 ants and then the number of ants which applied local search was then successively increased by one after a certain number of iterations. This variant only allows the iteration best ant to apply local search because, in this case, if all ants applied local search, the computation times would be too long.

8.4 Results

In this section, the computational results in terms of optimal parameters for MMAS, SA, GA and PSO algorithms are presented. The DFOS confidence interval (95%) for MMAS is also presented. Variations in α and β for the MMAS algorithm are discussed.

8.4.1 Optimal parameters for MMAS

Table 8.2 shows the average DFOS obtained by the $MMAS_{10+ls}$ algorithm for the studied case. The performance of the $MMAS_{10+ls}$ algorithm was tested against the CEM using equation (8.1) under the studied parameters.

Table 8.2: Average *DFOS for the $MMAS_{10+ls}$ algorithm (shading indicates the best result)

β	α	0.6	0.8	0.9	1	2	3	4	5	6	7	8	9	10
0	ρ=0.01	18.6	18.6	18.6	18.5	18.5	15.5	15.5	14.5	18.6	18.6	18.6	18.6	18.6
	ρ=0.02	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.9	18.9	18.9	18.9	18.9
	ρ=0.05	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.9	18.9	18.9	18.9
1	ρ=0.01	18.6	18.6	3.2	1.9	1.9	1.9	1.9	0.12	0.12	0.9	0.9	0.9	0.9
	ρ=0.02	18.6	18.6	3.2	3.2	3.2	0.9	0.9	0.12	0.12	0.9	0.9	0.9	0.9
	ρ=0.05	18.6	18.6	3.2	0.7	0.7	0.21	0.21	0.12	0.9	0.9	0.9	0.9	0.9
2	ρ=0.01	18.6	18.6	3.2	3.2	3.2	3.2	3.2	0.7	0.7	0.9	0.9	0.9	0.9
	ρ=0.02	18.6	18.6	3.2	3.2	3.2	0.9	0.9	0.12	0.12	0.9	0.9	0.9	0.9
	ρ=0.05	18.6	18.6	3.2	3.2	3.2	0.7	0.2	0.12	0.12	0.9	0.9	0.9	0.9
3	ρ=0.01	18.6	18.6	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	18.6	18.6	18.6
	ρ=0.02	18.6	18.6	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	18.6	18.6
	ρ=0.05	18.6	18.6	3.2	3.2	3.2	1.9	1.9	1.9	1.9	1.9	18.6	18.6	18.6
4	ρ=0.01	18.6	18.6	3.2	3.2	3.2	3.2	3.2	3.2	3.2	18.9	18.9	18.9	18.9
	ρ=0.02	18.6	18.6	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	18.9	18.9
	ρ=0.05	18.6	18.6	3.2	3.2	3.2	3.2	3.2	18.6	18.6	18.6	18.9	18.9	18.9
5	ρ=0.01	18.6	18.6	3.2	3.2	3.2	3.2	3.2	3.2	3.2	18.6	18.6	18.6	18.6
	ρ=0.02	18.6	18.6	3.2	3.2	3.2	3.2	3.2	3.2	18.9	18.9	18.9	18.9	18.9
	ρ=0.05	18.6	18.6	3.2	3.2	3.2	3.2	3.2	18.6	18.6	18.9	18.9	18.9	18.9

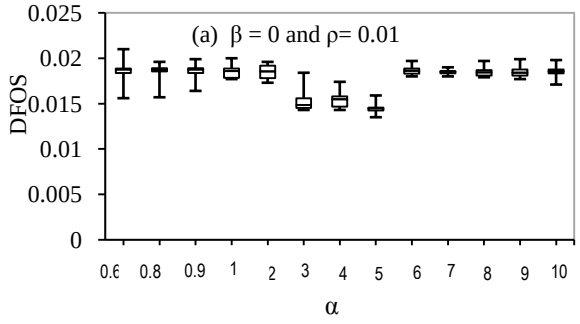
*Each DFOS × 0.001

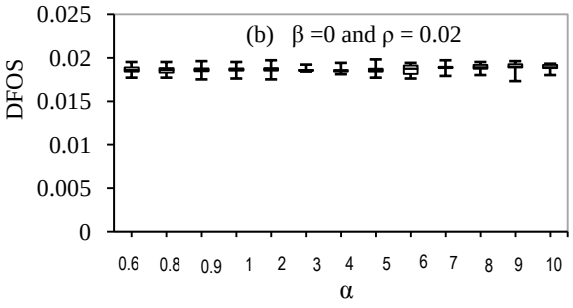
In the variant $MMAS_{10+ls}$, 10 ants were used and all ants were allowed to perform a local search. As described above, a mixed strategy between the iteration best ant and the global best ant was used to update the pheromone trails. The algorithm was executed per (α, β, ρ) parameter triplet for the 50 case studies. The best average DFOS obtained by $MMAS_{10+ls}$ are shaded in grey. It should be noted that this research was concerned with calculating the average DFOS; hence, the results presented in Table 8.2 are the average DFOS, not the optimal solution. The best combination of parameters for the $MMAS_{10+ls}$algorithm is shown in Table 8.3.

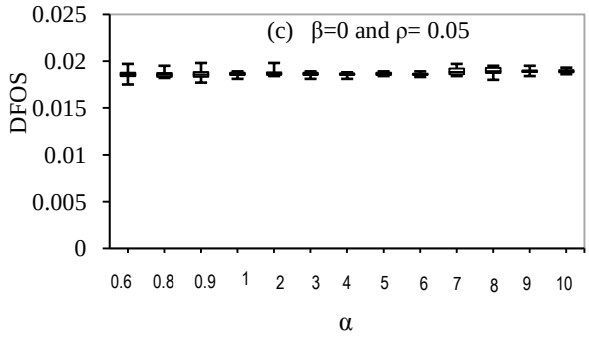
Table 8.3: Best combination of parameters for $MMAS_{10+ls}$

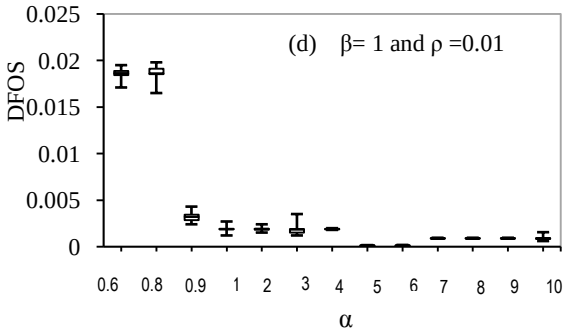
Parameters		
β	ρ	α
1	0.01	5,6
	0.02	5,6
	0.05	5
2	0.02	5,6
	0.05	5,6

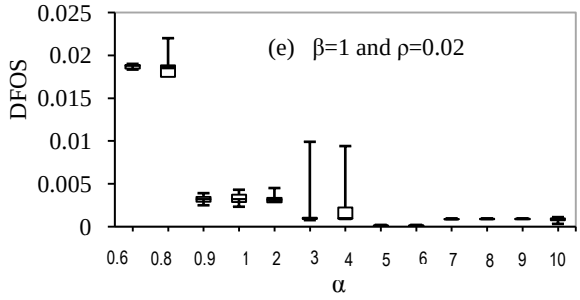
The graphics in Figure 8.1 are box-plots for different combinations of α, β and ρ. The box length gives an indication of the sample variability and the line across the box shows where the sample is centred. The position of the box in its whiskers and the position of the line in the box also tell us whether the sample is symmetric or skewed, either to the right or left. In Figure 8.1 (a, b and c) the parameters of MMAS are kept as standard values and the α parameter is varied within its specified range. It should be noted that when $\beta = 0$ it means that no heuristic information is used. The best result DFOS obtained is 0.0145. By increasing $\beta = 1$, as shown in Figure 8.1 (d, e and f), the range of α values that obtained best results are $\alpha = 5$, 6 and ρ=0.01. By increasing ρ to 0.02, the optimal range of α values remained the same. By increasing ρ to 0.05, the best average of DFOS is obtained when $\alpha = 5$.

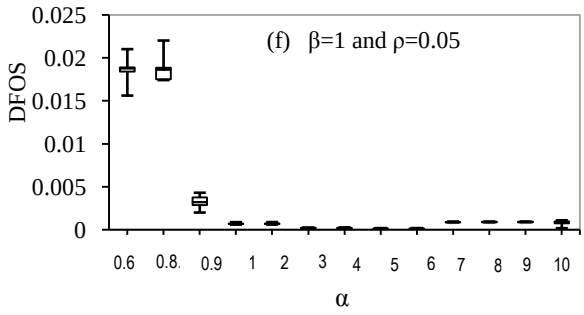

(a) β = 0 and ρ= 0.01
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α

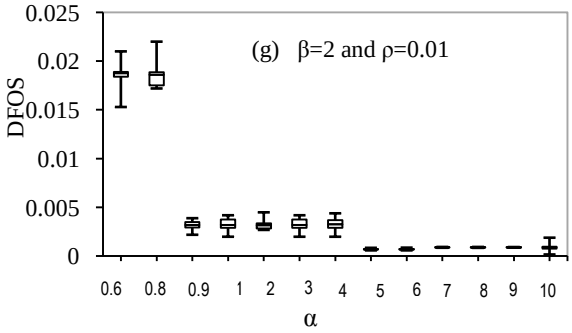

(b) β =0 and ρ = 0.02
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α

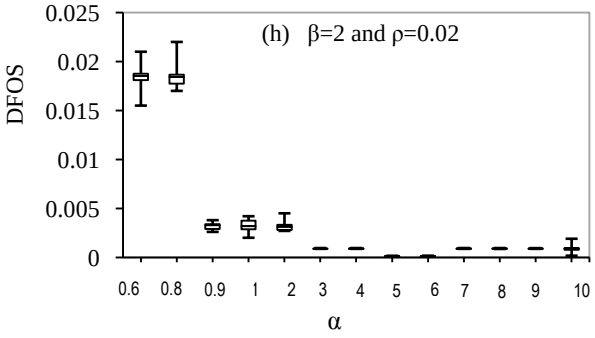

(c) β=0 and ρ= 0.05
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α

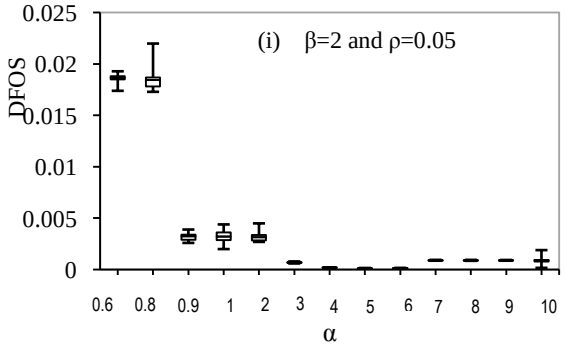

(d) β= 1 and ρ =0.01
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α

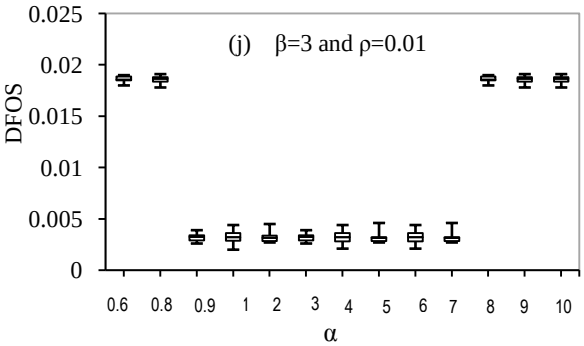

(e) β=1 and ρ=0.02
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α


(f) β=1 and ρ=0.05
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α


(g) β=2 and ρ=0.01
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α


(h) β=2 and ρ=0.02
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α


(i) β=2 and ρ=0.05
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α


(j) β=3 and ρ=0.01
DFOS
0.025
0.02
0.015
0.01
0.005
0
0.6 0.8 0.9 1 2 3 4 5 6 7 8 9 10
α

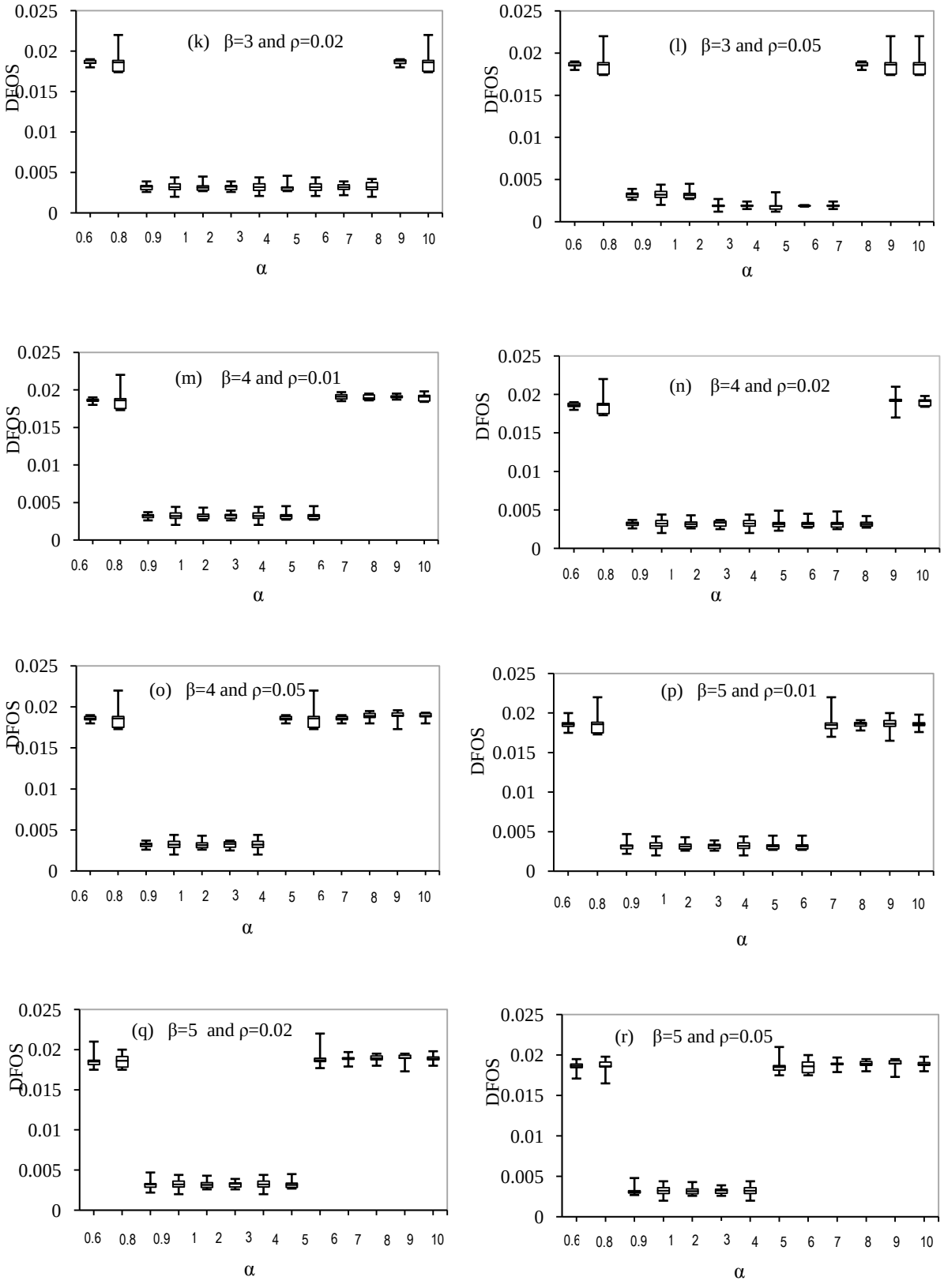


Figure 8.1: Influence of parameters α, β and ρ on the performance of MMAS-$_{10+ls}$ for the AOIS problem

In the case of $\beta = 2$, as shown in Figure 8.1 (g, h and i), the best results were obtained when ρ=0.01 and in the range of α values of 5 and 6.This situation was repeated by increasing ρ to 0.02 and 0.05. When β was increased to β =3, 4 and 5, as shown in Figures 8.1 (j–r), the performance of the $MMAS_{10+ls}$ became ineffective for obtaining good solutions. A big variation in the objective function value was observed when parameter α changed from 0.9 to 5. It can be seen that the performance of the MMAS algorithm is worse than in previous cases, except for $\beta = 2$. However, when β =3, 4 and 5 the variation in performance of MMAS is not very significant.

From the analyses above, one can see that parameters α, β and ρ are significant factors. From a sensitivity perspective, the parameters α and β are sensitive parameters. This is likely because they control the relative importance of the trail versus visibility, which relates to the essence of MMAS.

Table 8.4 shows the average DFOS for the same number of 50 cases conducted with the $MMAS_{+ls+ib}$ algorithm. In this variant, the algorithm started with a fixed number of 10 ants, and then the number of ants applying local searches was successively increased by one after a certain number of iterations (50). The aim was to obtain a good compromise between solution quality and computational speed. In this variant, only the best iteration ant was allowed to apply the local search. A mixed strategy between the best iteration ant and the best global ant was used to update the pheromone trails. As shown in Table 8.4, the best averages of DFOS obtained by $MMAS_{+ls+ib}$ are shaded grey. The best results were obtained with β=1, ρ=0.01 and in the range $\alpha = 7$ and 8. By increasing ρ=0.02 and 0.05 the range of α values which obtained the best results decreased to $\alpha = 5$ and 6. In the case of β =2 the best results were obtained with $\alpha = 5$ and 6 and ρ=0.02. By increasing ρ to 0.05 the best results are obtained with $\alpha = 6$.

Table 8.4: Average DFOS for the $MMAS_{+ls+ib}$ algorithm (shading indicates the best result)

β	α	0.6	0.8	0.9	1	2	3	4	5	6	7	8	9	10
0	ρ=0.01	18.6	18.6	18.6	18.5	18.5	15	15	14.5	18.6	18.6	18.6	18.6	18.6
	ρ=0.02	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.9	18.9	18.9	18.9
	ρ=0.05	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.6	18.9	18.9	18.9	18.9
1	ρ=0.01	3.2	3.2	3.2	1.9	1.9	1.9	1.9	1.9	1.9	0.21	0.21	4.2	4.2
	ρ=0.02	3.2	3.2	3.2	3.2	3.2	0.7	0.7	0.12	0.12	0.21	0.21	4.1	4.1
	ρ=0.05	3.2	3.2	3.2	0.72	0.72	0.21	0.21	0.12	0.12	0.19	0.19	4.1	4.1
2	ρ=0.01	18.9	3.2	3.2	3.2	3.2	3.2	3.2	0.7	0.7	1.2	1.5	2.1	4.1
	ρ=0.02	18.9	3.2	3.2	3.2	3.2	1.3	0.7	0.15	0.15	1.5	1.5	2.4	4.1
	ρ=0.05	18.9	3.2	3.2	3.2	3.2	0.7	0.2	0.21	0.15	1.2	1.5	2.4	4.1
3	ρ=0.01	5.2	5.2	5.2	5.2	5.2	5.2	5.2	3.2	3.2	3.2	18.6	18.6	18.6
	ρ=0.02	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	3.2	3.2	3.2	18.6	18.6
	ρ=0.05	5.2	5.2	5.2	5.2	5.2	4.1	4.1	4.1	4.1	4.1	18.6	18.6	18.6
4	ρ=0.01	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	18.9	18.9	18.9	18.9
	ρ=0.02	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	18.9	18.9
	ρ=0.05	5.2	5.2	5.2	5.2	5.2	5.2	5.2	18.6	18.6	18.6	18.9	18.9	18.9
5	ρ=0.01	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	18.6	18.6	18.6	18.6
	ρ=0.02	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	18.9	18.9	18.9	18.9	18.9
	ρ=0.05	5.2	5.2	5.2	5.2	5.2	5.2	5.2	18.6	18.6	18.9	18.9	18.9	18.9

*Each DFOS × 0.001

The best combinations of parameters for the $MMAS_{+ls+ib}$ algorithm are presented in Table 8.5.

Table 8.5: Best combinations of parameters for $MMAS_{+ls+ib}$

Parameters		
β	ρ	α
1	0.01	7,8
	0.02	5,6
	0.05	5,6
2	0.02	5,6
	0.05	6

The results for different combinations of α, β and ρ are plotted in Figure 8.2 to determine the relationship between these three parameters. Figure 8.2 shows influence of each parameter α, β and ρ when using the variant $MMAS_{+ls+ib}$ algorithm. The performance of $MMAS_{+ls+ib}$ is very similar to the previous $MMAS_{10+ls}$ algorithm. However the best results were obtained when β =1 and 2 and $\alpha = 5$ and 6 as shown in Figure 8.2 (b, c). As β >2 the performance of the MMAS algorithm worsens. It can be observed that the parameters α, β and ρ are significant factors in the MMAS algorithm. Also, it can be observed that the parameters α and β are sensitive parameters. This is likely because they control the relative importance of the pheromone trails versus heuristic information, which relates to the essence of MMAS algorithm.

8.4.2 Confidence intervals

The confidence interval (CI) provides a range that is highly likely (95% or 99%) to contain the parameter being estimated. The 95% CI is defined as "a range of values for a variable of interest constructed so that this range has a 95% probability of including the true value of the variable" (Attia, 2005). Due to sampling, the point estimate is probably not identical to the population parameter. Table 8.6 shows the 95% DFOS CIs for the $MMAS_{10+ls}$ algorithm. The results are based on the same general cost model case study. As a general rule, the narrower the CI the better it is. In this chapter, the aim is to tune the optimal parameters to find the

minimum total cost for the AOIS problem. As a result, the best parameters are those that produce the lowest DFOS. In Table 8.6 the best results corresponding to the best parameters are shaded in grey. These parameters produced the narrowest CI.

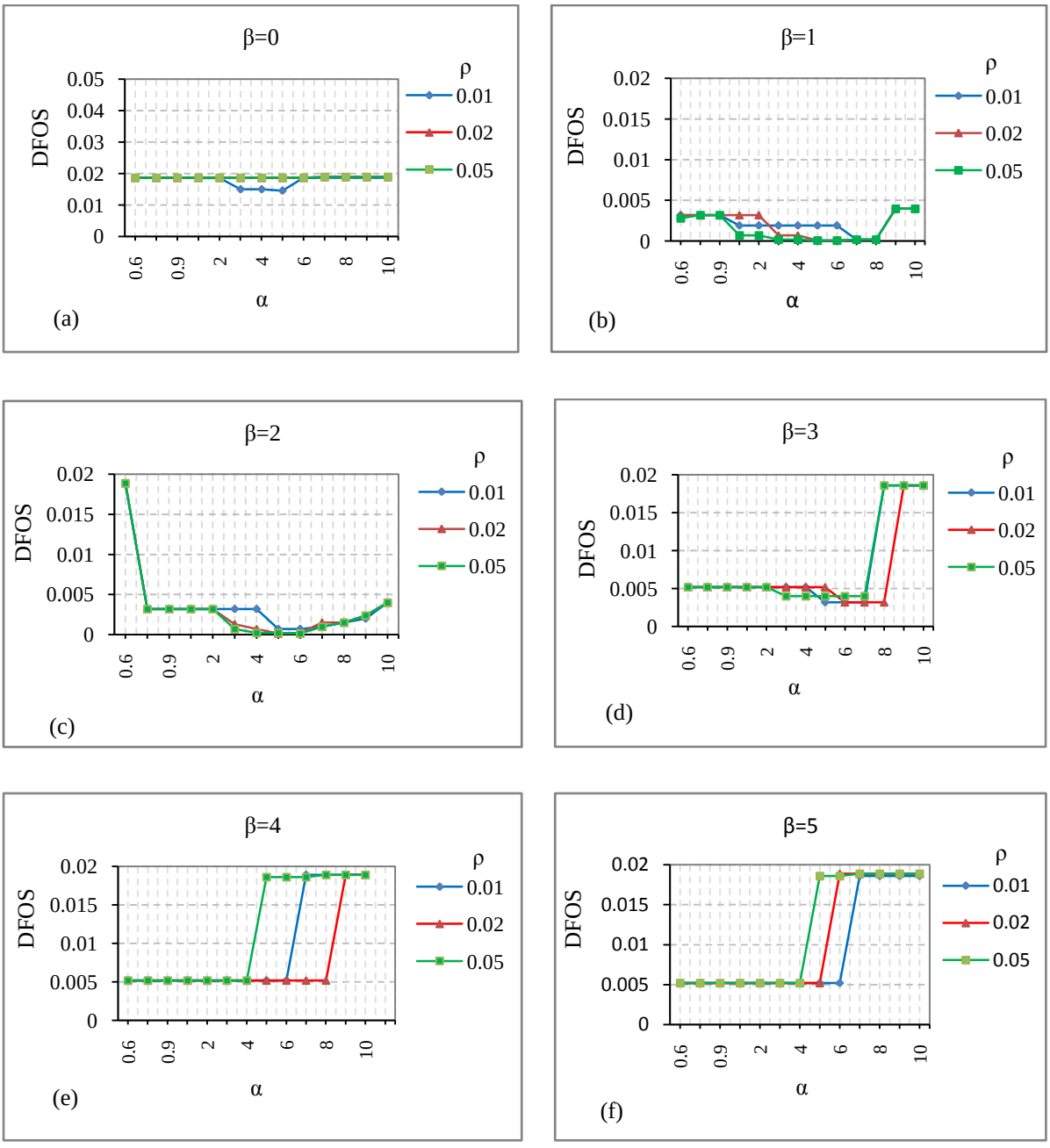


Figure 8.2: Influence of parameters α, β and ρ on the performance of $MMAS_{+ls+ib}$ for the AOIS problem

Table 8.6: 95% DFOS confidence intervals for $MMAS_{10+ls}$

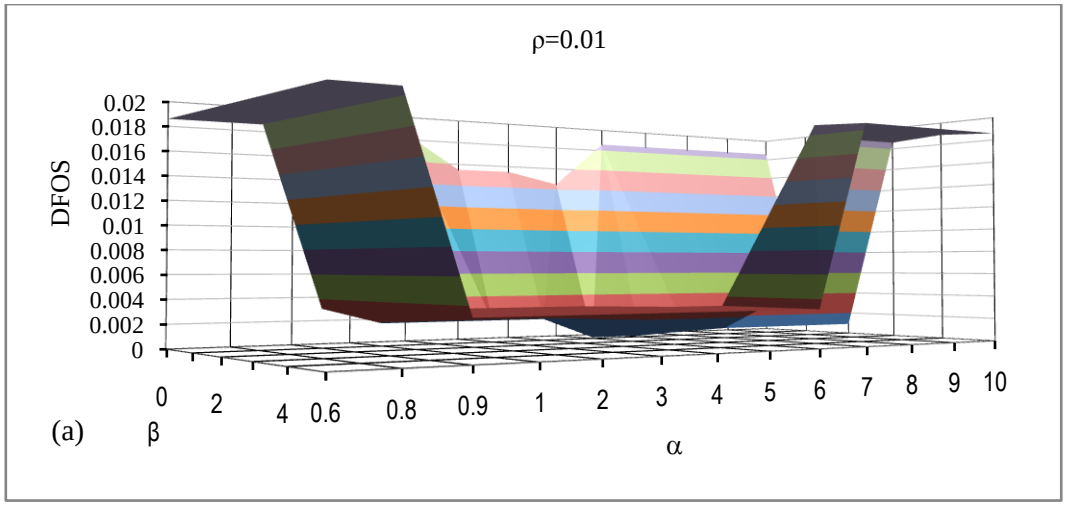
	α	0.6	0.8	0.9	1	2	3	4	5	6	7	8	9	10
β=0	ρ=0.01	(18.45,18.74)	(18.46,18.73)	(18.45,18.74)	(18.33,18.66)	(18.35,18.64)	(15.01,15.98)	(15.00,15.99)	(14.36,14.63)	(18.46,18.73)	(18.46,18.73)	(18.45,18.74)	(18.44, !8.75)	(18.51,18.68)
	ρ=0.02	(18.48,18.71)	(18.47,18.72)	(18.48,18.71)	(18.48,18.71)	(18.48,18.71)	(18.47,18.72)	(18.48,18.71)	(18.48,18.71)	(18.72,19.07)	(18.72,19.07)	(18.74,19.05)	(18.74,19.05)	(18.74,19.05)
	ρ=0.05	(18.49,18.70)	(18.47,18.72)	(18.48,18.71)	(18.49,18.70)	(18.48,18.71)	(18.49,18.70)	(18.48,18.71)	(18.47,18.72)	(18.48,18.71)	(18.75,19.04)	(18.75,19.04)	(18.75,19.04)	(18.75,19.04)
β=1	ρ=0.01	(18.45,18.74)	(18.46,18.73)	(3.10, 3.28)	(1.80, 1.99)	(1.81, 1.98)	(1.81, 1.98)	(1.81, 1.98)	(0.095,0.104)	(0.095,0.104)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)
	ρ=0.02	(18.45,18.74)	(18.46,18.73)	(3.12, 3.27)	(3.11, 3.287)	(3.12, 3.27)	(0.892,0.907)	(0.892,0.907)	(0.095,0.104)	(0.095,0.104)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)
	ρ=0.05	(18.45,18.74)	(18.46,18.73)	(3.10, 3.29)	(0.69, 0.70)	(0.69, 0.70)	(0.187,0.212)	(0.187,0.212)	(0.088,0104)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)	(0.891,0.908)
β=2	ρ=0.01	(18.45,18.74)	(18.46,18.73)	(3.11, 3.28)	(3.11, 3.28)	(3.11, 3.28)	(3.10, 3.29)	(3.11, 3.28)	(0.691,0.708)	(0.691,0.708)	(0.89,0.907)	(0.89,0.907)	(0.89,0.908)	(0.89,0.908)
	ρ=0.02	(18.45,18.74)	(18.46,18.73)	(3.11, 3.28)	(3.11, 3.28)	(3.11, 3.28)	(0.89, 0.908)	(0.89, 0.907)	(0.094,0.105)	(0.094,0.105)	(0.89,0.907)	(0.89,0.907)	(0.89,0.908)	(0.89,0.908)
	ρ=0.05	(18.45,18.74)	(18.46,18.73)	(3.11, 3.29)	(3.10, 3.29)	(3.11, 3.28)	(0.69, 0.708)	(0.19, 0.206)	(0.094,0.105)	(0.094,0.105)	(0.89,0.906)	(0.89,0.908)	(0.89,0.908)	(0.89,0.908)
β=3	ρ=0.01	(18.45,18.74)	(18.46,18.73)	(3.12, 3.29)	(3.11, 3.28)	(3.12, 3.29)	(3.11, 3.28)	(3.10, 3.27)	(3.11, 3.28)	(3.10, 3.27)	(3.11, 3.27)	(18.45,18.74)	(18.44,18.75)	(18.51,18.68)
	ρ=0.02	(18.45,18.74)	(18.46,18.73)	(3.12, 3.29)	(3.11, 3.28)	(3.12, 3.29)	(3.11, 3.28)	(3.11, 3.28)	(3.10, 3.27)	(3.12, 3.29)	(3.11, 3.28)	(3.12, 3.29)	(18.45,18.76)	(18.45,18.70)
	ρ=0.05	(18.45,18.74)	(18.46,18.73)	(3.10, 3.27)	(3.11, 3.28)	(3.12, 3.29)	(1.80, 1.98)	(1.81, 1.98)	(1.81, 1.98)	(1.80, 1.98)	(18.0, 19.8)	(18.5, 18.6)	(18.4, 18.7)	(18.4, 18.75)
β=4	ρ=0.01	(18.45,18.74)	(18.46,18.73)	(3.12, 3.29)	(3.10, 3.27)	(3.11, 3.28)	(3.10, 3.27)	(3.11, 3.28)	(3.10, 3.27)	(18.7, 19.05)	(3.10, 3.27)	(18.74,19.05)	(18.74,19.05)	(18.74,19.05)
	ρ=0.02	(18.45,18.74)	(18.46,18.73)	(3.11, 3.28)	(3.12, 3.29)	(3.12, 3.29)	(3.11, 3.28)	(3.11, 3.28)	(3.10, 3.27)	(18.7, 19.05)	(3.10, 3.27)	(18.74,19.05)	(18.74,19.05)	(18.74,19.05)
	ρ=0.05	(18.45,18.74)	(18.46,18.73)	(3.11, 3.28)	(3.10, 3.29)	(3.11, 3.28)	(3.10, 3.27)	(3.12, 3.29)	(18.48,18.71)	(18.48,18.71)	(18.48,18.71)	(18.75,19.05)	(18.74,19.05)	(18.75,19.05)
β=5	ρ=0.01	(18.45,18.74)	(18.46,18.73)	(3.10, 3.27)	(3.12, 3.29)	(3.11, 3.28)	(3.10, 3.29)	(3.10, 3.27)	(3.12, 3.29)	(3.12, 3.29)	(18.46,18.73)	(18.45,18.74)	(18.44,18.75)	(18.51,18.68)
	ρ=0.02	(18.45,18.74)	(18.46,18.73)	(3.11, 3.28)	(3.11, 3.28)	(3.11, 3.28)	(3.12, 3.29)	(3.10, 3.27)	(3.11, 3.28)	(18.74,19.05)	(18.75,19.04)	(18.76,19.06)	(18.75,19.04)	(18.75,19.04)
	ρ=0.05	(18.45,18.74)	(18.46,18.73)	(3.11, 3.28)	(3.10, 3.27)	(3.12, 3.29)	(3.11, 3.28)	(3.11, 3.28)	(18.48,18.71)	(18.48,18.71)	(18.72,19.07)	(18.74,19.06)	(18.74,19.05)	(18.74,19.05)

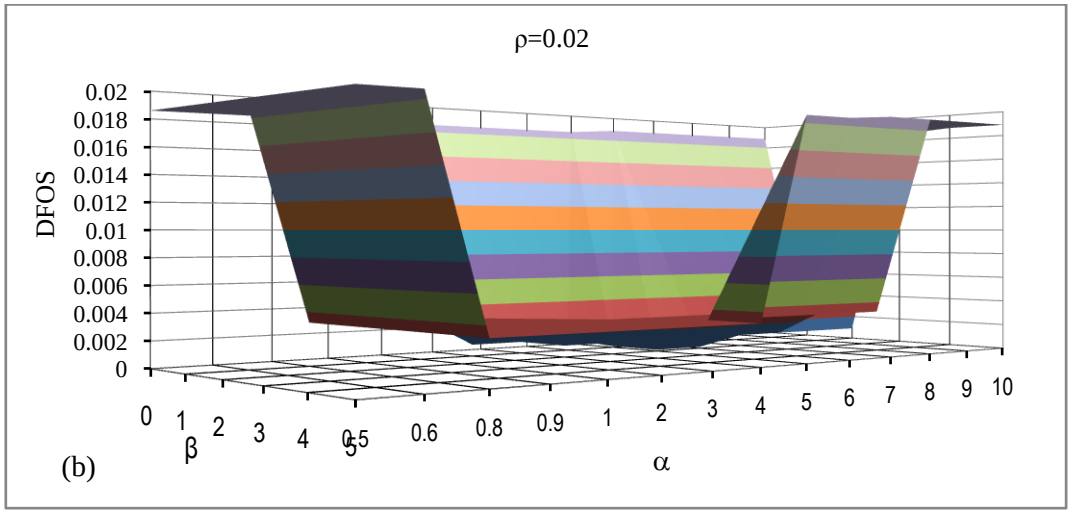
*Each confidence interval (lower, upper) × 0.001

8.4.3 Variations in α and β

Different values were compared for the MMAS parameters α and β on the same general cost model case study. These two parameters determine the mutual influence of the trail strength (τ) and the heuristic information (η) on the choice of the next workstation. As can be seen from Figure 8.3 the parameter settings of α and β have a considerable influence on the performance of the MMAS.In the experiments conducted the evaporation rate is set to 0.01, 0.02 and 0.05, as shown in Figure 8.3 (a, b and c). The results obtained in this experiment are consistent with our understanding of the algorithm: a high value of α means the trail is very important and therefore ants tend to choose workstations chosen by other ants in the past. On the other hand, low values of α make the algorithm very similar to a stochastic greedy algorithm. It can be observed that when $\alpha >> \beta$ (e.g. $\alpha = 8$ and β=1) then the algorithm will make decisions based mainly on the information learned, as represented by the pheromone. This is true until the value of β becomes very high (e.g. β=4 or 5), in this case even if there is a high amount of trail on an edge, an ant always has a high probability of choosing another workstation that has a higher desirability. In other words, if β >2 (e.g. β=4) the algorithm will act as a greedy heuristic algorithm, mainly selecting workstations that have the maximum cost (higher desirability) and disregarding the impact of these decisions on the final solution quality. As a result the algorithm does not find very good solutions in the number of cycles used in the experiment. However, when $\alpha < 5$ the pheromone strength loses its dominant status during the searching process and becomes ineffective in guiding the ant towards good solutions regions.

High values of β and/or low values of α make the algorithm very similar to a stochastic greedy algorithm. It should be noted that the parameter ρ is present in the pheromone update rule. The evaporation rate is vital in the ability of the algorithm to explore different solutions and to avoid becoming trapped in a local optimum.





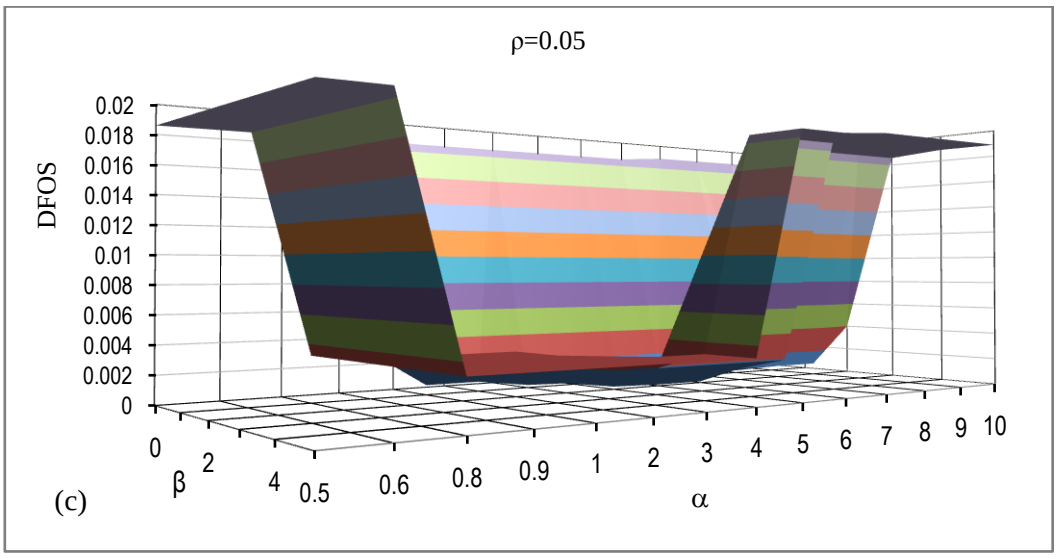


Figure 8.3: Analysis of $MMAS_{10+ls}$ algorithm

A high value of evaporation rate allows much greater exploration but can lead to early convergence, as will be described in more detail the following sections. In this example, 5 inspection points are chosen out of the 12 available, so over 41 % of the processes/inspections receive an increase in pheromone. In the problem under consideration, the best result is obtained for parameter combinations β =1, 2 and $\alpha = 5, 6$ as indicated by the lowest point in blue colour which represents the lowest total cost as shown in Figure 8.3.

Table 8.7 shows the 95% DFOS CIs for the $MMAS_{+ls+ib}$ algorithm. The results are also based on the same general cost model case study. It can be seen from Table 8.7 the best results corresponding to the best parameters are shaded with grey colour. The same general cost model case study is applied to the $MMAS_{+ls+ib}$ to study the influence parameters α and β. In the conducted experiments the evaporation rate is set ρ= 0.01, 0.02 and 0.05 as shown in Figure 8.4 (a, b and c). It can be seen from Figure 8.4 that the variation between α and β in the $MMAS_{+ls+ib}$ is very similar to the previous $MMAS_{10+ls}$ algorithm which is described above in this section. In the problem under consideration, the best result is obtained for parameter combinations β =1, 2 and $\alpha = 5$ as indicated by the lowest point in blue which represents the lowest total cost as shown in Figure 8.4.

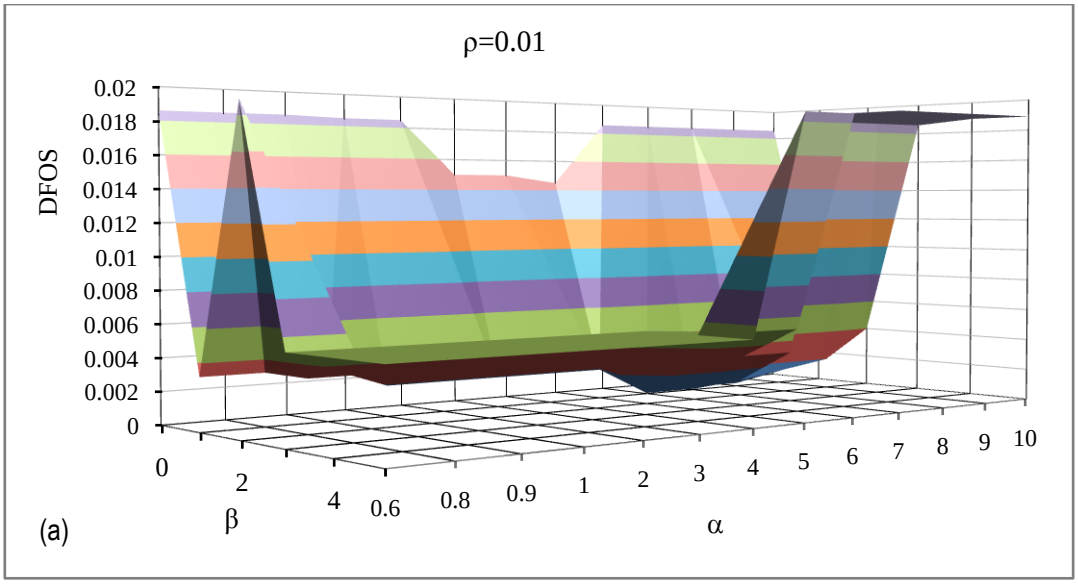
8.5 Tuning of genetic algorithm parameters

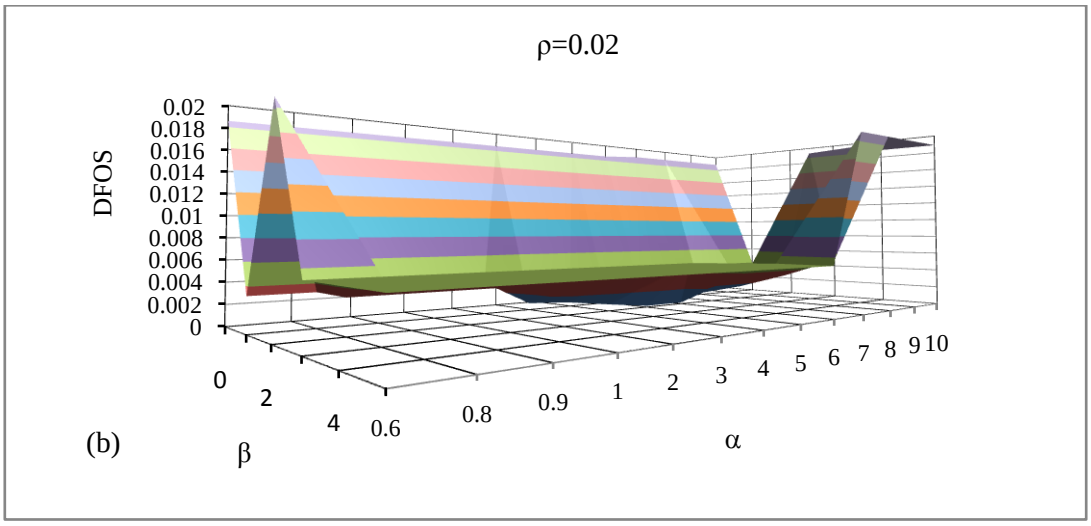
Genetic algorithms (GA) are inspired by models of natural evolution of species and use the principle of natural selection, which favours individuals that are more adapted to a specific environment for survival and further evolution. Each individual in an evolutionary algorithm typically represents a solution with an associated fitness value. Parameter values of GA must be tuned in advance in order to fully specify a complete GA. Inappropriate parameter values may not be able to solve problems effectively.

Table 8.7: 95% DFOS confidence intervals for $MMAS_{+ls+ib}$

	α	0.6	0.8	0.9	1	2	3	4	5	6	7	8	9	10
β=0	ρ=0.01	(18.44,18.75)	(18.44,18.75)	(18.44,18.75)	(18.22,18.66)	(18.33,18.66)	(15.02,15.97)	(15.00,15.99)	(14.35,14.62)	(18.44,18.75)	(18.45,18.74)	(18.45,18.74)	(18.45,18.75)	(18.50,18.66)
	ρ=0.02	(18.47,18.72)	(18.47,18.72)	(18.48,18.71)	(18.48,18.71)	(18.48,18.71)	(18.47,18.72)	(18.48,18.71)	(18.48,18.71)	(18.72,19.07)	(18.72,19.07)	(18.74,19.05)	(18.74,19.05)	(18.74,19.05)
	ρ=0.05	(18.48,18.71)	(18.47,18.73)	(18.48,18.71)	(18.50,18.71)	(18.50,18.71)	(18.49,18.70)	(18.47,18.72)	(18.47,18.72)	(18.48,18.71)	(18.74,19.05)	(18.75,19.04)	(18.75,19.04)	(18.75,19.04)
β=1	ρ=0.01	(3.24, 3.35)	(3.23, 3.34)	(3.24, 3.35)	(1.80, 1.99)	(1.81, 1.98)	(1.81, 1.98)	(1.81, 1.98)	(1.81, 1.98)	(1.81, 1.98)	(0.188, 0.21)	(0.188, 0.21)	(3.97, 4.02)	(3.97,4.02)
	ρ=0.02	(3.24, 3.35)	(3.10, 3.29)	(3.12, 3.27)	(3.11, 3.28)	(3.12, 3.27)	(0.893,0.908)	(0.892,0.907)	(0.096,0.105)	(0.096,0.105)	(0.188, 0.21)	(0.188, 0.21)	(3.97, 4.02)	(3.97, 4.02)
	ρ=0.05	(3.11, 3.28)	(3.10, 3.29)	(3.10, 3.29)	(0.68, 0.708)	(0.69, 0.707)	(0.187,0.212)	(0.187,0.212)	(0.096,0.106)	(0.097,0.107)	(0.192,0.20)	(0.191,0.208)	(3.96,4.01)	(3.96, 4.01)
β=2	ρ=0.01	(18.80,18.95)	(3.10, 3.29)	(3.11, 3.28)	(3.11, 3.28)	(3.10, 3.29)	(3.10, 3.29)	(3.11, 3.28)	(0.691,0.706)	(0.691,0.708)	(0.965,1.03)	(1.47,1.52)	(2.48,2.51)	(3.96, 4.01)
	ρ=0.02	(18.8,18.95)	(3.10, 3.29)	(3.11, 3.28)	(3.11, 3.28)	(3.11, 3.28)	(0.89, 0.908)	(0.89, 0.907)	(0.094,0.105)	(0.094,0.105)	(1.47,1.52)	(1.47,1.52)	(2.48,2.51)	(3.97, 4.02)
	ρ=0.05	(18.8,18.95)	(3.11, 3.28)	(3.10, 3.29)	(3.10, 3.29)	(3.12, 3.27)	(0.69, 0.709)	(0.19, 0.206)	(0.097,0.106)	(0.094,0.105)	(0.96,1.03)	(1.47,1.52)	(2.48,2.51)	(3.97, 4.02)
β=3	ρ=0.01	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(3.10, 3.29)	(3.10, 3.29)	(3.11, 3.28)	(18.46,18.73)	(18.46,18.73)	(18.46,18.73)
	ρ=0.02	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(3.12, 3.29)	(3.11, 3.28)	(3.12, 3.29)	(18.45,18.76)	(18.45,18.70)
	ρ=0.05	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(3.97, 4.02)	(3.97, 4.02)	(3.97, 4.02)	(3.97, 4.02)	(3.97, 4.02)	(18.55, 18.6)	(18.44, 18.7)	(18.4, 18.75)
β=4	ρ=0.01	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(18.75,19.04)	(18.75,19.04)	(18.74,19.05)	(18.74,19.05)
	ρ=0.02	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(18.74,19.05)	(18.74,19.05)
	ρ=0.05	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(18.48,18.71)	(18.48,18.71)	(18.48,18.71)	(18.75,19.05)	(18.74,19.7)	(18.75,19.05)
β=5	ρ=0.01	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(18.46,18.73)	(18.45,18.74)	(18.44,18.75)	(18.51,18.68)
	ρ=0.02	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(18.74,19.05)	(18.75,19.04)	(18.76,19.06)	(18.75,19.04)	(18.75,19.04)
	ρ=0.05	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(5.16, 5.23)	(18.48,18.71)	(18.48,18.71)	(18.72,19.07)	(18.74,19.05)	(18.74,19.05)	(18.74,19.05)

*Each confidence interval (lower, upper) × 0.001





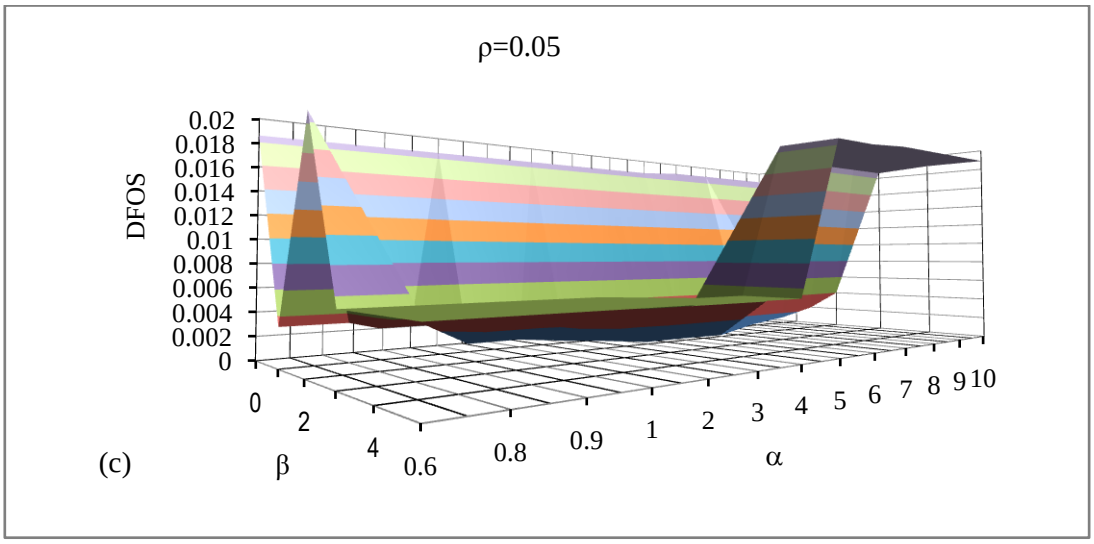


Figure 8.4: Analysis of MMAS$_{+ls+ib}$ algorithm

On the other hand, finding the optimal setting will increase the overall search capability and better solutions can be found. The parameter settings are based on general recommendations found in textbooks on GA/EA's (Michaelewicz and Fogel, 2000, Reeves, 1993), on an insight into the problem specifics, as suggested by Silver (2004), and on several exploratory test runs.

8.5.1 Summary of GA parameters

There are three parameters that affect the performance of a GA: population size, crossover probability and mutation probability.

Population size: is the number of individuals used in one generation. GA's with a low size population are prone to premature convergence. This is because a GA has very few possibilities to perform a crossover and only a small part of the search space is explored. On the other hand, if the population size is large, it slows down the convergence rate of the algorithm and increases the number of objective function evaluations.

Crossover probability: is the percentage of individuals in the mating pool able to perform recombination or exchange. Crossover is a continuous variable within [0, 1]. If there is no crossover, the offspring is an exact copy of its parents. If there is crossover, the offspring is made from parts of the parents' individuals. If the crossover probability is 100%, then all offspring are made by crossover. Crossover introduces new solution strings into the population and searches for better strings. However it is good to allow a part of the population to survive in the next generation.

Mutation probability: is a unary operator, aimed at introducing random modifications in an individual. Mutation is a continuous variable within [0, 1]. If there is no mutation, the offspring carries crossover information or is an exact copy of the parents, however without any change in the information. If mutation occurs the chromosome changes, if the mutation probability is 100%, the whole chromosome will change. A low mutation rate may result in a

loss of some important characteristics in a population. Mutation is used to prevent the falling of the GA into local extreme, although it should not occur very often as then the GA will in fact change into a random search (Yuan and Gallagher, 2005).

In summary, different studies pointed out that population size, crossover probability and mutation rate are the most influential parameters in the performance of a GA (Smit and Eiben, 2011, Van Volsem, 2007). A method to optimise these parameters is presented in this chapter.

8.5.2 GA parameter settings

The values chosen for the GA parameters are representative of the range of values typically seen in the literature (Smit and Eiben, 2011, Sadegheih, 2007, Van Volsem, 2007).The application of the GA to the general cost model are repeated here for each of the combinations of the following parameter levels:

Population size: number of individuals (inspection plans) used in one generation. The values: low [20, 30, 40], medium [50, 60, 70] and high [80, 90, 100] of this parameter were tested for the tuning of the algorithm.

Crossover rate: the values [0.10, 0.20, 0.60, 0.8 and 0.90] were tested for the tuning of the algorithm.

Mutation rate: the values [0.001, 0.002, 0.003, 0.004, 0.005, 0.01, 0.017 and 0.02] were tested for the tuning of the algorithm.

8.5.3 Results

The same general cost model case study is used to test the GA, which consists of 50 different problems. Different parameter combinations of the GA were then applied for each of these combinations in order to calculate the average DFOS. The aim is to tune the parameters of the GA. The results obtained by the method are compared against the optimal results obtained

by the CEM. The aim is to measure the DFOS for the results obtained by the GA. The optimal parameters are those that can obtain the minimum DFOS. Table 8.8 shows the best combinations of the GA parameters. The results from different parameter combinations of the GA tuning are presented in Appendix D.

Table 8.8: Optimal GA parameters

Population	Cross over	Mutation rate	Average DFOS
80	0.8	0.005	0.0012

8.6 Tuning simulated annealing parameters

Simulated Annealing (SA) is a stochastic search method, emulative of the physical annealing process, in which an alloy is cooled gradually so that a minimal energy state is achieved. Just like other optimisation methods, the algorithm of simulated annealing consists of operating parameters that should be set appropriately in order to achieve the best performance. The most efficient cooling schedule may be found through trial and error, and by observing the effect on both the quality of the resulting solution and the rate at which the process converges (Demirkol et al., 2001). The parameter settings were based on the general recommendations of Guo and Zheng (2005), Sadegheih (2007) and Kolahan and Abachizadeh (2010).

8.6.1 Summary of SA parameters

The cooling schedule of a SA algorithm consists of four components.

Starting temperature: the initial temperature acts as the state of the system at the beginning of the optimisation procedure. In general, the initial temperature is set in a way that allows the acceptance of most transitions in the neighbourhood moves. In other words, the Boltzmann distribution acceptance criterion should be approximately equal to 1 at the starting temperature, $\exp(-\Delta C/T_0) \cong 1$ where ΔC is the difference between the amount of the objective function of the current and candidate solutions (Guo and Zheng, 2005).

Temperature decrement: is the most important variable in the entire SA algorithm. The method in which the temperature parameter is reduced is an important feature of this algorithm. At every iteration k the temperature is decreased according to the formula, $T_{k+1} = \alpha \cdot T_k$ where the parameter α, usually called the cooling rate, is such that $(0 < \alpha < 1)$ (Kolahan and Abachizadeh, 2010). The cooling schedule is the way in which the temperature is decremented and is critical to the success of the algorithm.

Iterations at each temperature: possibly the second most important factor that determines the computational effort required to run a particular cooling schedule for simulated annealing is the number of transitions in one temperature step. The number of iterations at each temperature is chosen so that the system is sufficiently close to the stationary distribution at that temperature (Suman and Kumar, 2006). Enough iterations at each temperature should be performed if the temperature is to be periodically decreased. If too few iterations are performed, all represented states will not be searched and the solution will not be able to reach the global optimum.

Final temperature: it is common to let the temperature decrease until it reaches zero, however this can make the algorithm run for a long time, especially when a geometric cooling schedule is being used. In practice it is not necessary to let the temperature reach zero because as it approaches zero the chances of accepting a worse move are almost the same as the temperature being equal to zero. Therefore the stopping criteria can either be a suitably low temperature or the moment at which the system is "frozen" at the current temperature (i.e. no better or worse moves are accepted) (Kolahan and Abachizadeh, 2010). The maximum number of iterations is another approach. The maximum number of subsequent iterations allowed without any improvement is also another appropriate criterion.

In summary, the temperature decrement and number of trials per temperature step have the greatest influence on the simulated annealing algorithm's performance. The most efficient

cooling schedule may be found through trial and error and by observing the effect on both the quality of the resulting solution and the rate at which the process converges (Demirkol, 2001). A method to optimise these parameters is presented in this chapter.

8.6.2 SA parameter settings

The values chosen for SA parameters are representative of the range of values typically seen in the literature (Kolahan and Abachizadeh, 2010, Sadegheih, 2007). The application of the SA to the general cost model is repeated here for each of the combinations of the following parameter levels:

Temperature decrement: the values [0.4, 0.5, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9 and 0.95] were used to tune the algorithm.

Iterations at each temperature: the values [100, 300, 500, 800 and 1200] were used to tune the algorithm.

8.6.3 Results

The same general cost model case study is used to test the SA algorithm. Different parameter combinations of the SA were then applied to the cases for each of these combinations in order to calculate the average DFOS. The aim is to tune parameters of the SA. The results obtained by the method are then compared against the optimal results obtained by the CEM. The aim is to measure the DFOS for the results obtained by the SA. The optimal parameters are those that can obtain the minimum DFOS. Table 8.9 shows the best combinations of the SA parameters. The results from different parameter combinations of SA tuning are presented in Appendix E.

Table 8.9: Optimal SA parameters

Temperature decrement	Iterations at each temperature	Average DFOS
0.85	300	0.0007

8.7 Tuning particle swarm optimisation parameters

Particle swarm optimisation (PSO) is a population-based stochastic optimisation technique inspired by the social behaviour of birds flocking or fish schooling. Particle swarm optimisation (PSO) is a population-based stochastic optimisation technique inspired by the social behaviour of birds flocking or fish schooling. Selecting the best parameters inertia weight, cognitive coefficient, and number of particles (ω, c_1, c_2, M) for PSO is another model selection task. Several empirical and theoretical studies have been performed on the parameters of PSO from which useful information can be obtained (Clerc and Kennedy, 2002, Kennedy and Mendes, 2002, Ozcan and Mohan, 1999, Reyes and Coello, 2006, Shi and Eberhart, 1998, 1999). In this section, a simulation of various parameter settings under different conditions for defining the best parameter combination for the AOIS problem is presented.

8.7.1 Summary of PSO parameters

The most influential parameters in the performance of PSO are the cognitive coefficients, c_1, c_2, inertia weight ω and number of particles (Aliabadi et al., 2011, Hsieh et al., 2007, Rezazadeh et al., 2009).

Inertia weight: inertia weight ‘‘ω’’ was not mentioned by Eberhart and Kennedy (1995), but was introduced by Shi and Eberhart (1998). Its goal is to control the impact of the past velocity of a particle over the current one, influencing the local and global exploration abilities of the algorithm. Shi and Eberhart found that a PSO with an inertial weight in the range of 0.8 to 1.2 has, on average, a better performance; that is it has a greater chance of finding the global optimum within a reasonable number of iterations. On the other hand, a large value of ω keeps particles at high velocity and prevents them from becoming trapped in local optima. A small value of ω maintains particles at a low velocity and encourages them to exploit the same search area.

Cognitive coefficients: c_1 and c_2 are cognitive coefficients and are both constants between zero and one. Thus the particle flies through potential solutions toward the local best (*lbest)* and the global best (*gbest*) in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optima. The cognitive coefficients c_1 and c_2 represent the weightings that pull each particle toward *lbest* and *gbest*. Low values let particles wander around their local neighbourhood, while high values cause particles to fly towards, or past, optimal solutions (Yin et al., 2006).Different problems will have different values, and these range from1 to 4 (Hsieh et al., 2007,Shuang et al., 2011,Yin et al., 2006).

Number of particles: in a PSO the population is the number of particles in a problem space. Particles are initialised randomly. Each particle will have a fitness value, which will be evaluated by a fitness function to be optimised in each generation. In the population or swarm, more particles may increase the success of searching for optima due to more thorough sampling of the state space. However, more particles require more evaluation runs, leading to a higher optimisation time cost.

8.7.2 PSO parameters settings

The values chosen for PSO algorithm parameters are representative of the range of values typically seen in the literature (Aliabadi et al., 2011**,** Hsieh et al., 2007, Liao et al., 2007). The application of PSO to the general cost model is repeated here for each of the combinations of the following parameter levels:

Inertia weight (ω): four levels (0.8, 0.9, 1.1 and 1.2) are used to test the PSO algorithm.

Cognitive coefficients (c_1 and c_2): six levels for each coefficient are used to test the PSO algorithm (1, 1.5, 2, 2.5, 3 and 3.5). In these experiments the parameters c_1 and c_2 are tested by making them equal and unequal, and making c_1 larger than c_2, and vice versa.

Number of particles: different sizes of particles are used to test the PSO algorithm (10, 20, 40, 60, 80 and 100).

8.7.3 Results

The same general cost model case study is used to test the PSO algorithm. Different parameter combinations of the PSO algorithm were then applied to the cases for each of these combinations in order to calculate the average DFOS. The aim is to tune the parameters of the PSO algorithm. The results obtained by the algorithm are then compared against the optimal results obtained by the CEM. The aim is to measure the DFOS for the results obtained by the PSO algorithm. The optimal parameters are those that can obtain the minimum DFOS. Table 8.10 shows the best combinations of the PSO algorithm parameters. The results from different parameter combinations of PSO tuning are presented in Appendix F.

Table 8.10: Optimal GA parameters

Inertia weight	Number of particles	c_1	c_2	Average DFOS
1.1	80	2	2	0.0007

8.8 Conclusion

The optimal combinations of the most influential parameters were identified for the MMAS algorithm, SA, GA and PSO. These algorithms were applied to the same general cost model case study in order to calculate the average DFOS. As a result, the optimal parameters can then be obtained to tackle the AOIS problem. It was found that these optimal parameters led to a significantly improved performance of the developed algorithms. For a specific problem the optimal parameters will be slightly different, but these results suggest that these parameters should be suitable for many AOIS problems with a similar structure. Early indications show that MMAS algorithm outperforms the other relevant algorithms in terms of solution quality. Two different variants of the MMAS algorithm were developed. The two variants were $MMAS_{10+ls}$, and $MMAS_{+ls+ib.}$ The experimental results confirmed that the performance of the MMAS, GA, SA and PSO algorithms depends on the appropriate setting

of the parameters. Following the tuning of the parameters for the MMAS algorithm, in the next chapter the behaviour of the MMAS algorithm will be investigated. Furthermore, the best parameters obtained for the SA, GA and PSO algorithms will be used to test performance of the MMAS algorithm in chapter 10.

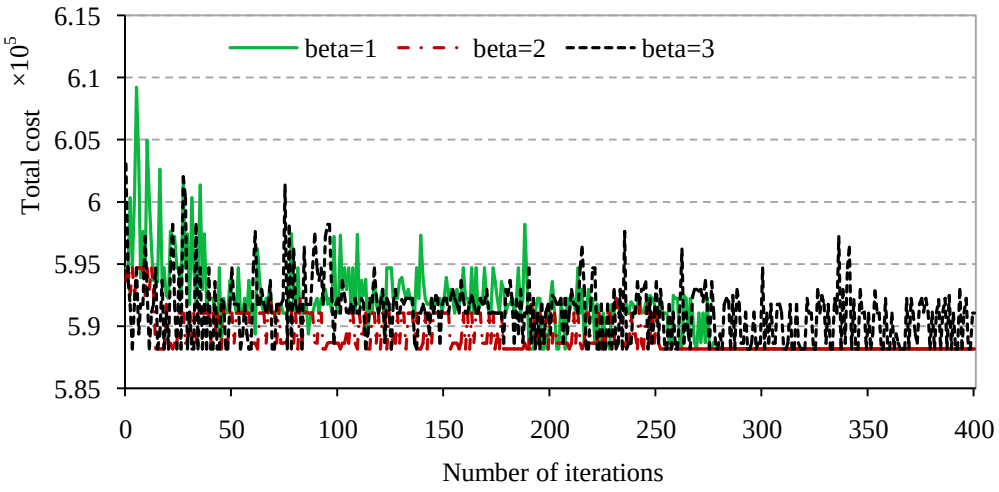
Chapter 9
Behaviour of MMAS and sensitivity analysis

In this chapter, the behaviour of MMAS and the sensitivity of the control parameters for the AOIS problem are studied. Different variants of the MMAS algorithm are tested. The convergence of the MMAS algorithm is given through the simulations of the general cost model case study. The importance of the local search and heuristic information for the MMAS algorithm are discussed. Global and iteration best versus a mixed strategy for updating pheromone trials are investigated. A stagnation measure is implemented to indicate the degree of diversity of the solutions constructed by the ants.

9.1 MMAS behaviour

In order to investigate the capability of the MMAS to solve the AOIS problem, the convergence of the algorithm and the impact caused by the parameter settings are investigated. To do so, nine sets of parameters [α=1, β=1,2,3, ρ=0.01] [α=2, β=1,2,3, ρ=0.01] [α=5, β=1,2,3, ρ=0.01] [α=1, β=1,2,3, ρ=0.02] [α=2, β=1,2,3, ρ=0.02] [α=5, β=1,2,3, ρ=0.02] [α=1, β=1,2,3, ρ=0.05] [α=2, β=1,2,3, ρ=0.05] [α=5, β=1,2,3, ρ=0.05] were tested in the same general cost model case study. In Figures 9.1 to 9.9, the x-axis stands for 400 iteration steps and the y-axis is the value of the objective function (total cost). These figures illustrate the fact that parameter variations strongly affect the speed of the convergence of the objective function. Under the parameter settings of α=1, β=1 and ρ=0.01, as shown in Figure 9.1, convergence occurred within 270 iteration steps. As β increased to 2, the convergence occurred early within 175 iteration steps. However there was no convergence within the 400 iteration steps with parameter settings of α=1, β=3 and ρ=0.01.

Under parameter settings of α=2, β=1, 2, 3 and ρ=0.01, as shown in Figure 9.2, convergence occurred early within 125 iteration steps with β=1. By increasing β=2 and 3 the convergence occurred within 175 iteration steps and within 300 iteration steps respectively.



.

Figure 9.1: Comparing the convergence of MMAS under parameter settings $\alpha = 1, \rho = 0.01$

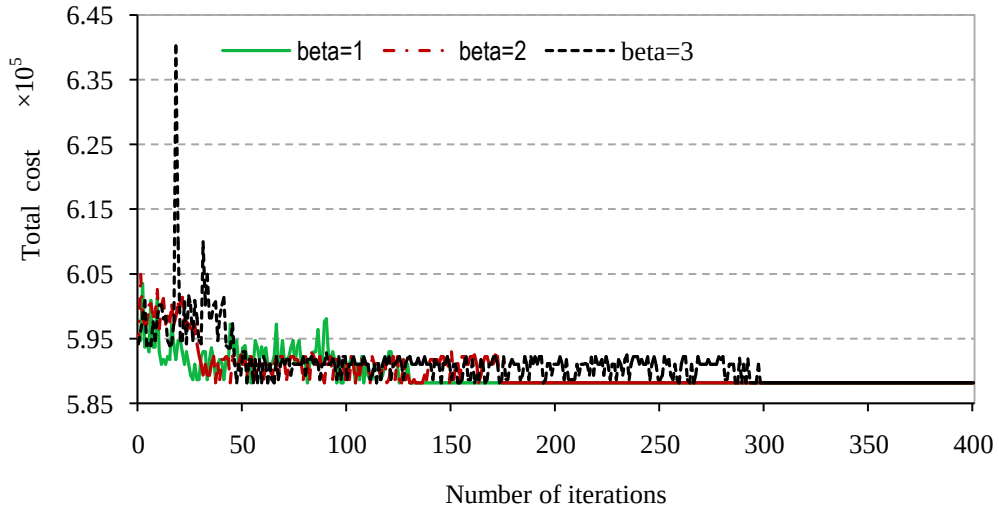


Figure 9.2: Comparing the convergence of MMAS under parameter settings $\alpha = 2, \rho = 0.01$

By increasing α to 5, as shown in Figure 9.3, convergence occurred earlier, within 50 iteration steps when β=1 and in 150 iteration steps when β=2 and 3. It can be observed from Figures 9.1 to 9.9 that as α and ρ increased and β decreased convergence occurred early. The results showed that the ratio between α and β was the key deriver to the convergence speed of the results. Since α and β are the parameters of relative influence of the pheromone strength and

the heuristic information respectively, the relative influence of the pheromone strength, which dominated the searching process, makes the convergence occur early.

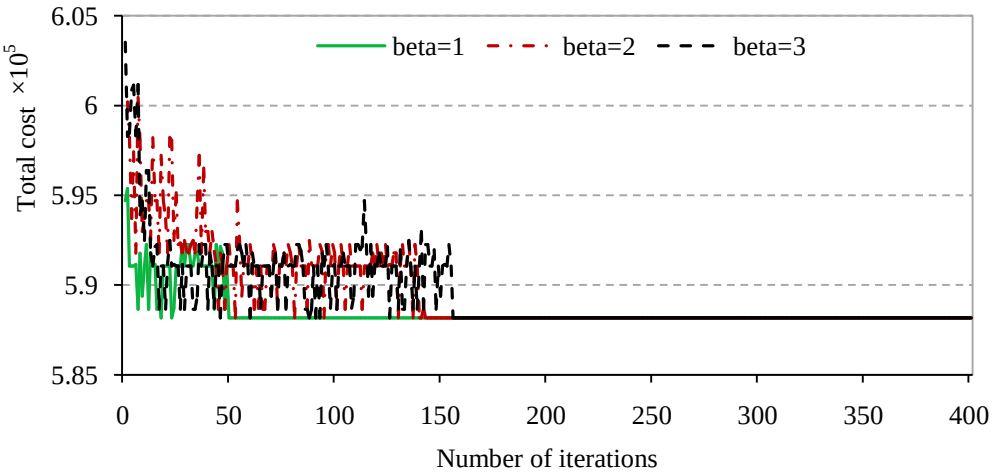


Figure 9.3: Comparing the convergence of MMAS under parameter settings $\alpha = 5, \rho = 0.01$

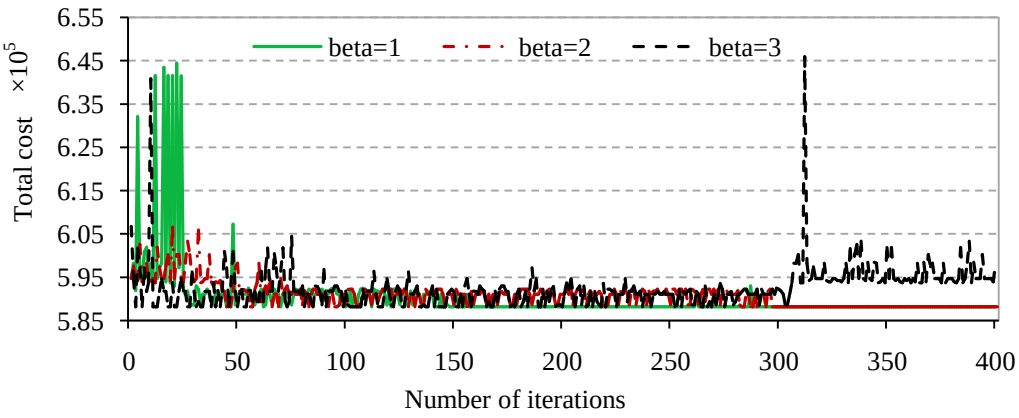


Figure 9.4: Comparing the convergence of MMAS under parameter settings $\alpha = 1, \rho = 0.02$

With the status of the pheromones on the searching trails dominated, the heavily accumulated pheromones would strongly bias the choice of ants. In such a manner, more and more subsequent ants would choose the same trail, and consequently convergence would occur. Conversely, when the pheromone strength lost its dominant status during the searching process, i.e. the ratio between α and β became smaller and smaller, the influence of pheromone strength would not be used as the indicator of choice. In this circumstance, the convergence speed would decrease, and convergence would be delayed.

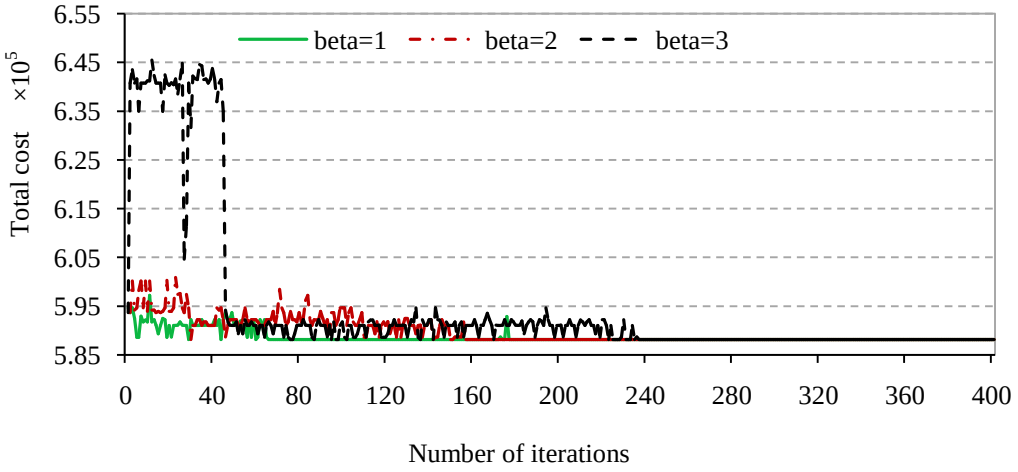


Figure 9.5: Comparing the convergence of MMAS under parameter settings $\alpha = 2, \rho = 0.02$

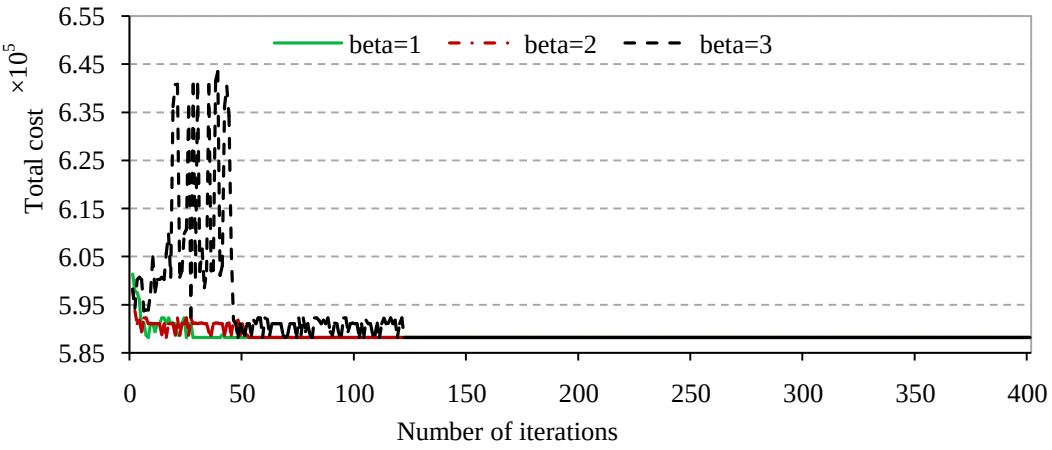


Figure 9.6: Comparing the convergence of MMAS under parameter settings $\alpha = 5, \rho = 0.02$

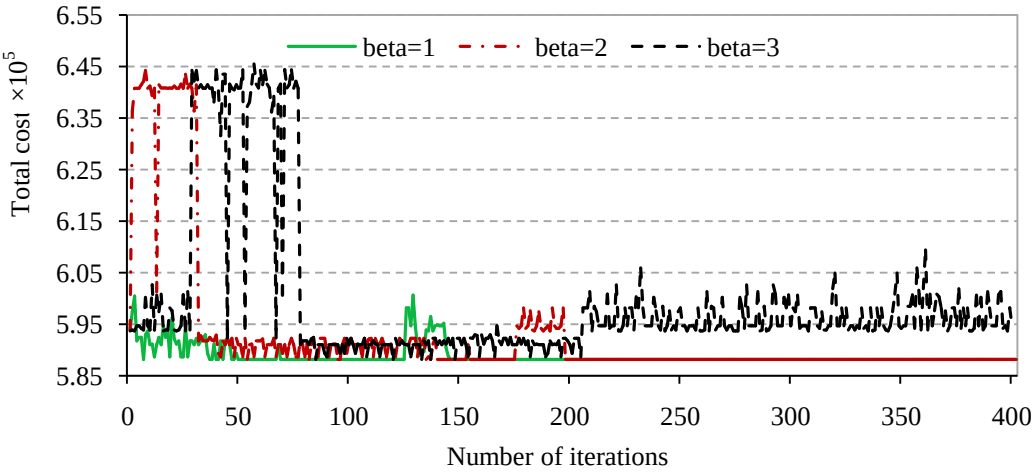


Figure 9.7: Comparing the convergence of MMAS under parameter settings $\alpha = 1, \rho = 0.05$

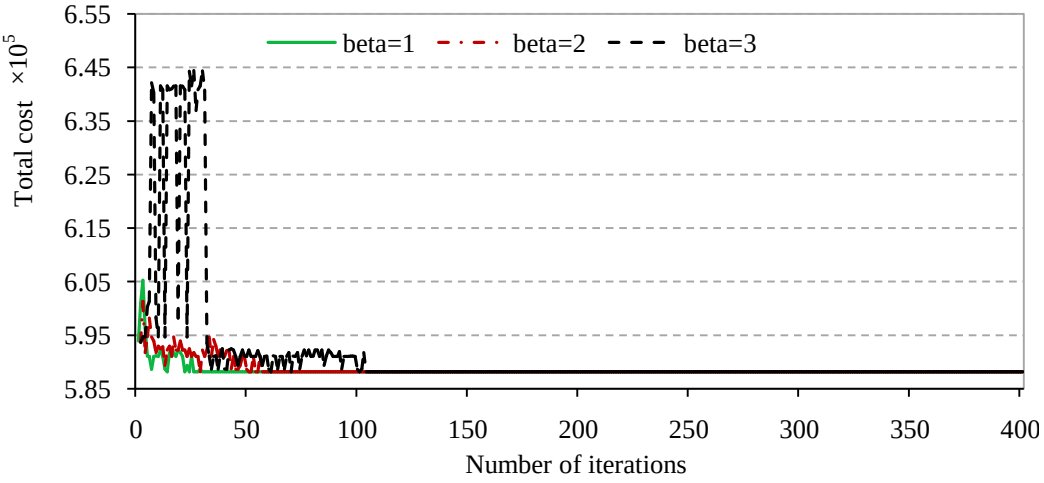


Figure 9.8: Comparing the convergence of MMAS under parameter settings $\alpha = 2, \rho = 0.05$

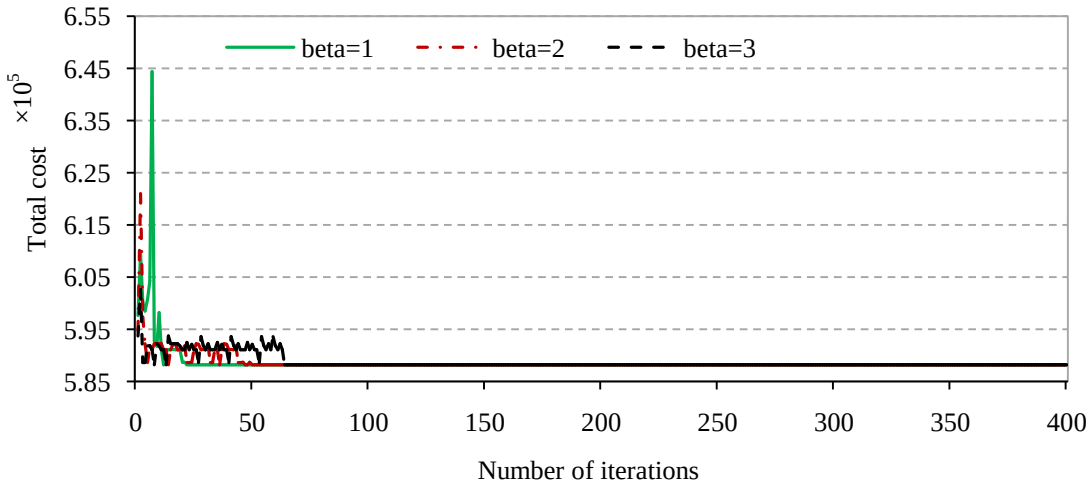


Figure 9.9: Comparing the convergence of MMAS under parameter settings $\alpha = 5, \rho = 0.05$

From Figures 9.1 to 9.9, the convergence speeds under parameter settings α=5 and β=1, were faster than those with parameter settings of α=1 and β=3. The larger the ratio between α and β, the faster the convergence speed under the parameter ρ=0.05. Thus the convergence speed under parameters α=5 and β=1 occurred earlier than that under parameters α=1 and β=3. Correspondingly, there was no convergence when $\beta > \alpha$. It can be seen from Figures 9.4 and 9.7 that when the algorithm converged to the optimal solution, the pheromone trails were initialised to the maximum pheromone trails (τ_{max}) after a certain number of iteration steps

(300 and 200 iterations respectively). The aim is to increase exploration of the search space. It can be observed from both Figures 9.4 and 9.7 that the ants continue to explore a subset of the search space as computation proceeds. In fact this feature is one of the features of the MMAS as described in chapter 6. In summary, the ratio between α and β is identified as the key deriver to convergence speed using MMAS. The results showed that the larger the ratio between α and β, the faster the speed of convergence.

Figure 9.10 depicts the evolution curves of MMAS with ρ varying from 0.01 to 0.1, α =5, β =1, λ =0.05 and the other parameters are kept at standard values. The curves are for different values of ρ and are averages of 50 independent executions. It can be observed from Figure 9.10 that a better convergence speed is obtained when larger values of ρ are used. However, with increasing ρ, the best values of different curves will increase and the search may stop within less iterations. This is due to the fact that larger pheromone evaporation ratios on arcs will accelerate pheromone evaporation and, hence, the search concentrates earlier around the best trails seen so far.

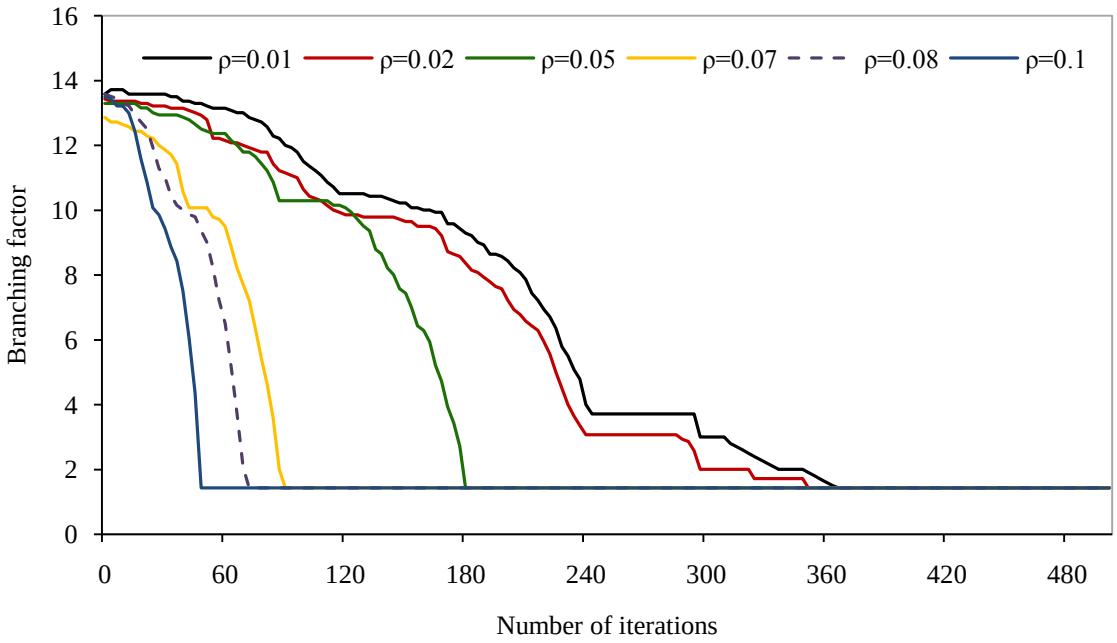


Figure 9.10: Evolution curves of MMAS for different values of ρ

If ρ is large it is easy to reach marked relative differences between the pheromone trails on arcs contained in high quality trails, and those which are not part of the best trails in a few iterations, so the algorithm may stagnate and prematurely converge. Otherwise, for a lower ρ the pheromone trails on arcs which do not belongs to the high quality trails will not decrease faster, and the algorithm will be able to explore a wider search space, although longer evolution iterations will be needed. Therefore, if larger total evolution iteration is used, a lower ρ can be selected for obtaining a better converging value; otherwise a higher ρ will be helpful for a better convergence speed.

9.2 Sensitivity analysis

The vast majority of papers reviewed did not consider sensitivity analysis. Sensitivity analysis was only performed on models to test the optimal solution in the models developed by Yum and McDowell (1987) and Parak et al. (1988).This was most probably because their models were developed using Integer Linear Programming, and hence they included it easily. As the MMAS algorithm relies on a number of user-defined parameters that control its behaviour, sensitivity analysis for the most influential parameters α, β and ρ was conducted to study the importance of each parameter for the AOIS problem considered. The purpose of the sensitivity analyses was to obtain a detailed understanding of the impact of each of the user-defined MMAS parameters on the algorithms' searching behaviour. To achieve this, each parameter was varied over a specified range, while all other parameters were maintained at their 'standard' values.

It is well known that the number of workstations is one of the most important characteristics for the AOIS problem. In a serial multistage manufacturing process, as the problem size increases the number of inspection stations allocation possibilities increases exponentially in the search space size. To study the influence of the studied parameters on increasing the number of workstations, experimental studies are implemented on different sizes of AOIS

problem. Four different sizes of the AOIS problem are conducted. These sizes are 15, 16, 17 and 18 workstations. The number of feasible solutions generated from these workstations grows from 32,768 to 262,144. The aim is to determine the influence of MMAS parameters on the performance of the algorithm, particularly when the number of workstations increases. Each parameter was varied over a specified range, while all other parameters were kept at their 'standard' values.

Figure 9.11 shows the influence of parameter α on different sizes of the AOIS problem. Each point is the average of 50 runs of the MMAS algorithm. The value of α determines the relative importance of the pheromone level of the ant in the process of movement when guiding the ant colony search.

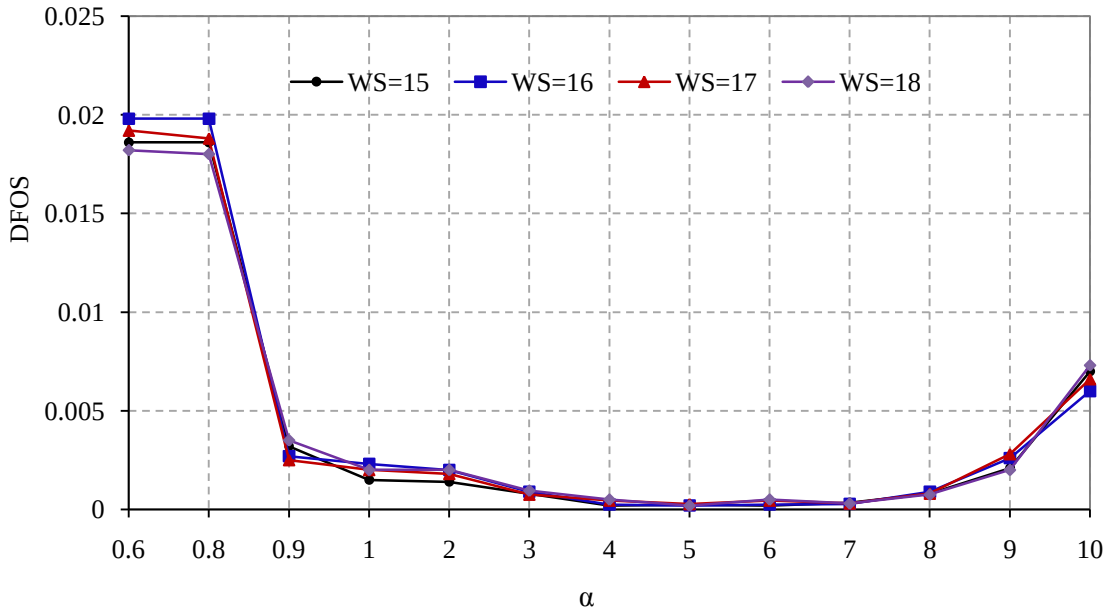


Figure 9.11: Sensitivity analysis of parameter α for different AOIS problems

It can be seen from Figure 9.11 that the best results are obtained when $2 < \alpha < 8$. However, the variation between these values is not so significant in terms of solution quality. On the other hand, when $\alpha < 1$, the algorithm shows poorer performance in terms of solution quality. This is because the pheromone strength in the algorithm is not enough to guide ants to the

best regions of the solution space. In this case, the influence of the pheromone strength is ineffective to guide ants to the good solutions. It is concluded that the parameter α is less sensitive within the specified range as the number of workstations significantly increases.

Figure 9.12 shows the influence of parameter β on different sizes of the AOIS problem. The value of β determines the relatively important degree of the heuristic information and the importance of the location of inspection stations relative to the workstations. It can be seen from Figure 9.12 that the best results are obtained with β values 1 and 2. However, the variation between these two values is not greatly significant in terms of solution quality. On the other hand, when $\beta = 0$ the algorithm shows a poorer performance. This situation is repeated when $\beta > 2$.

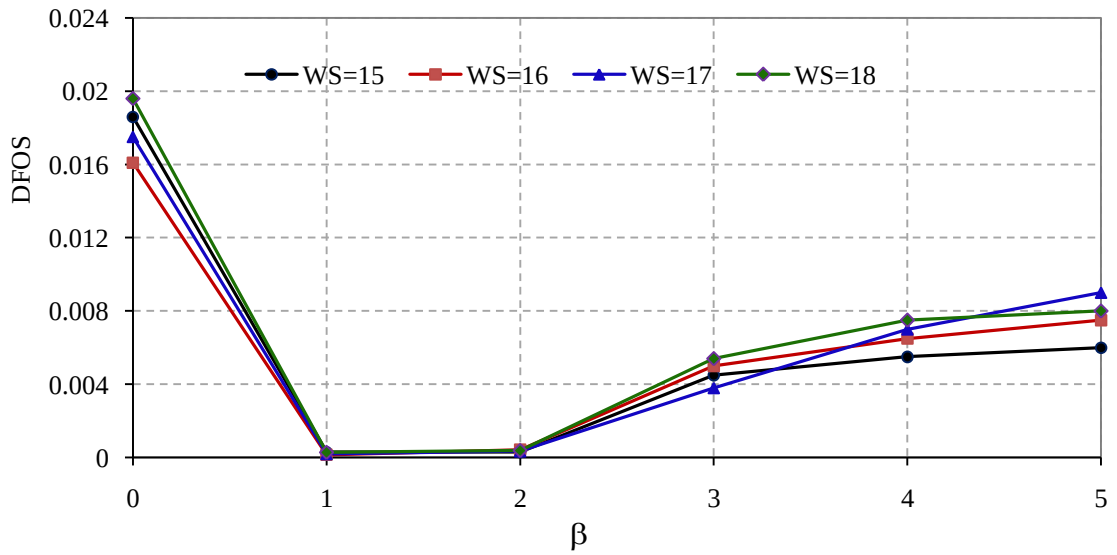


Figure 9.12: Sensitivity analysis of parameter β for AOIS problems

Figure 9.13 shows the influence of parameter ρ on different sizes of the AOIS problem. The aim is to represent different characteristics of manufacturing systems, in other words, to represent different AOIS problems. The value of ρ determines the evaporation speed of the pheromone trace. The evaporation rate is vital in the ability of the algorithm to explore different solutions and to avoid becoming trapped in a local optimum. The higher the value,

the faster the pheromone will evaporate and hence the faster the algorithm converges. On the other hand, using a very low value of ρ leads to more exploration of the MMAS algorithm and high quality solutions may be missed. Since the figure does not show significant differences between the values of 0.01 and 0.1, it can be stated that the model has a consistent behaviour for a broad range of values of ρ. At these values the best results can be obtained. These results are optimal values or near optima values. It is concluded that the parameter ρ is less sensitive to changes in the number of workstations.

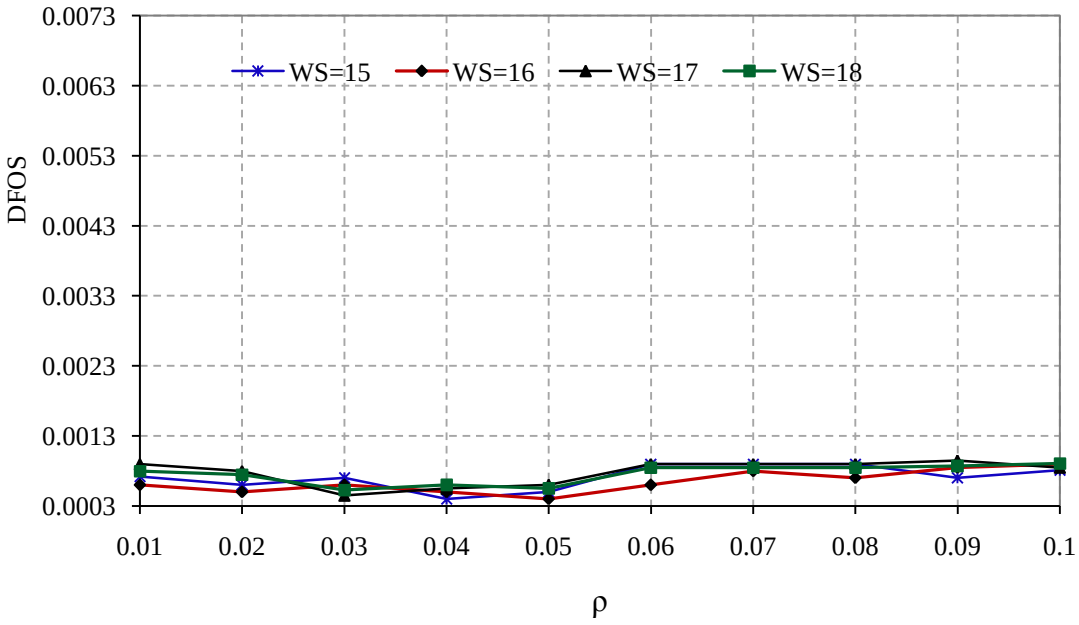


Figure 9.13: Sensitivity analysis of parameter ρ for AOIS problems

In summary, the experimental results confirmed that the studied parameters α, β and ρ do not have a significant influence on the performance of the MMAS algorithm as the number of workstation is increased.

9.3 Importance of the heuristic information

As described in chapter 6, two novel heuristic information methods were created to guide the ant to locate the inspection stations of workstations. These methods are operation cost, inspection cost, and defect rate method (OCDM) and Scores Method (SM). It should be noted

that there was no heuristic information created for the AOIS problem. In this section, the performance of the MMAS without using heuristic information (indicated by $MMAS_{-NH}$) is compared to its performance when using it. It should be noted that not using heuristic information is simply achieved by setting $\beta = 0$. The comparison of MMAS variants with and without using heuristic information is applied on the case study consisting of 15 workstations. The optimal solution for this number of workstations is known and all feasible solutions using CEM can be enumerated within a reasonable time. Experimental results are conducted for the MMAS algorithm with heuristic information ($MMAS_{-OCDM}$ and $MMAS_{-SM}$) and no heuristic information ($MMAS_{-NH}$). The results are based on the average of 50 different cases randomly generated from the general cost model case study; in other words, these cases represent 50 different problems. The experimental results for the MMAS algorithm with heuristic information ($MMAS_{-OCDM}$ and $MMAS_{-SM}$) and with no heuristic information ($MMAS_{-NH}$) in terms of the average number of optimal solutions, best average DFOS, worst average DFOS values and average processing time are presented in Table 9.1. The experimental results showed that the performance of the MMAS algorithm, in terms of solution quality when using heuristic information, was significantly better than without heuristic information. In particular, the OCDM heuristic information is better than the SM.

Table 9.1: Comparison of MMAS with and without heuristic information

Algorithm	Average number of optimal solutions (%)	Best average DFOS (not optimal)	Worst average DFOS	Average time (seconds)
$MMAS_{-NH}$	0	0.0186	0.52	12
$MMAS_{-OCDM}$	80	0.00015	0.0125	15
$MMAS_{-SM}$	74	0.00033	0.0103	17

The reason for this is that the OCDM type is based on three characteristics of the AOIS problem, operation cost, defect rate and inspection cost. The main aim of the heuristic information is to guide the ants through the assigning of inspection stations to workstations.

The workstations that have higher results have a higher probability to be selected by the ant. Placing inspection stations based on these characteristics lead to the detection of defective items before moving further to subsequent operations. As a result the total cost can then be minimised. On the other hand, the SM type of heuristic information is based on scores of operation cost and defect rate. This method takes scores of the operation cost and defect rate and use a combination of the two choices in one.

Figure 9.14 shows the convergence of MMAS toward near optimal solutions with and without using heuristic information. It is clear that in the case of $MMAS_{-NH}$ the ants are moving away from the near optimal solution. However, when using heuristic information in the cases of $MMAS_{-OCDM}$ and $MMAS_{-SM}$ the ants gradually attempted to follow the shortest path. As a result, the solution gradually moved towards the close-to-optimal solution, as the ants found almost similar paths.

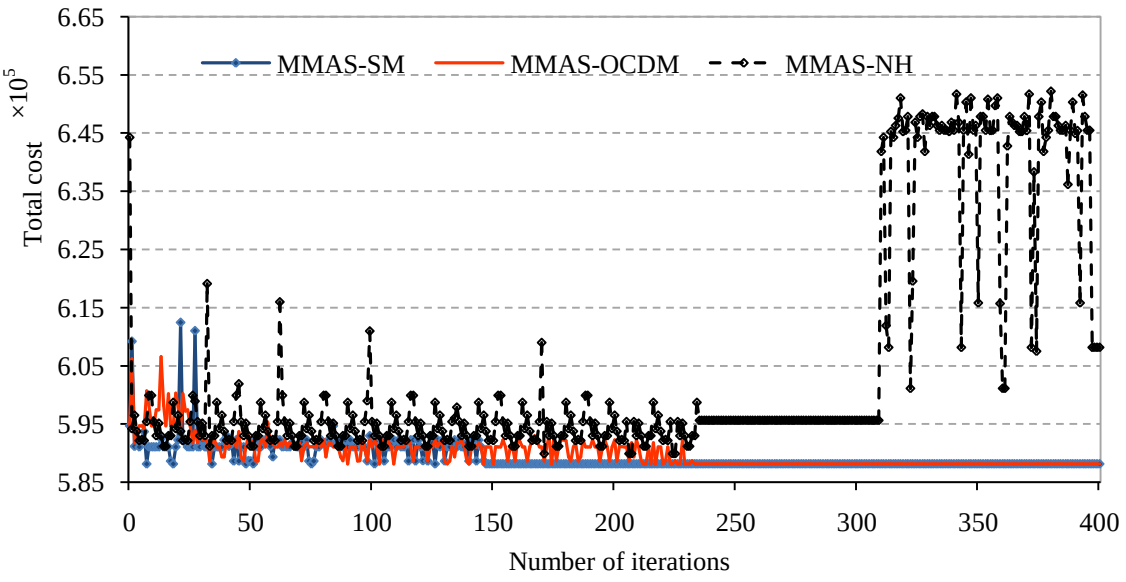


Figure 9.14: Comparing the convergence of MMAS with and without heuristic information

It is clear from the results that the heuristic plays a central role in guiding the algorithm and that a good heuristic applied strongly can produce accurate results. This is because using heuristic information makes the probable search space (the search space most likely to be

explored) becomes much smaller than the original search space. It is concluded that it is important to use heuristic information with the MMAS algorithm for tackling the AOIS problem.

9.4 Importance of local search

It has been proven that a local search applied to the solutions that the ants have constructed can improve the performance of an ant colony's optimisation algorithm. It should be noted that of the metaheuristic methods used in the literature review for tackling the AOIS problem, none of them used local search to improve the performance of their models. In this study, as described in chapter 6, six well-known neighbourhood structures are used to improve the performance of the MMAS algorithm. These are: crossover (CO), interchange(IC), swap (SW), single insertion (SI), delete and add (DA), and block insertion (BI). The MMAS algorithm is applied on different sizes of workstations with each of these local search methods. As described in chapter 6, to yield a further reduction in run-time and to focus the local search on promising regions where potential improvements can be found, don't look bits method is used. The comparison between them is based on the average deviation from the optimal solution and average CPU time. The results are based on the average of 50 different cases randomly generated for each problem size. The aim is to represent the different characteristics of the AOIS problem.

It can be seen from Table 9.2 that the local search methods developed have considerably improved the performance of the MMAS algorithm. This improvement can be observed in the average deviation from the optimal DFOS solution. In particular, the local search methods of crossover, single insertion and block insertion have better performance than the other methods in terms of solution quality. The superior results indicate the successful incorporation of the local search with the MMAS algorithm to escape local minimum points and increase the possibility of finding a better solution.

Table 9.2: Comparison of the local search effectiveness for the AOIS problem

WS	No local search		CO		IC		SW		SI		DA		BI	
	$DFOS_{avg}$	$time_{avg}$ (s)	$DFOS_{avg}$	$time_{avg}$ (S)	$DFOS_{avg}$	$time_{avg}$ (s)	$DFOS_{avg}$	$time_{avg}$ (s)	$DFOS_{avg}$	$time_{avg}$ (s)	$DFOS_{avg}$	$time_{avg}$ (s)	$DFOS_{avg}$	$time_{avg}$ (s)
12	0.0053	13	0.00012	11	0.0007	12	0.0008	12	0.00012	11	0.0008	12	0.00012	10
13	0.0043	15	0.00018	12	0.0008	14	0.0008	13	0.00015	12	0.0009	14	0.00014	11
14	0.0042	16	0.00017	14	0.0008	15	0.0009	14	0.00018	15	0.004	15	0.00016	14
15	0.0052	18	0.00018	16	0.0009	16	0.0009	18	0.00018	20	0.0009	17	0.00019	17
Average		16	0.00057	13	0.0008	14	0.00085	14	0.0005	15	0.00165	15	0.00052	13

This good performance can even be observed when the number of feasible solutions grows from 4,096, in the case of 12 workstations, to 32,768, in the case of 15 workstations. Regarding computation times, it is worth noting that in all cases the local search takes less time than the sum of the computation times necessary for the MMAS algorithm starting from an initial solution. This effect is caused by the good starting solution for the second local search and the fewer number of improvement steps necessary to reach a local optimum. Applying the ant algorithm to problems of different sizes makes a difference in terms of speed. Generally, the bigger the problem the longer it takes to find good, very good or optimal solutions.

Figures 9.15 and 9.16 are box-plots showing the performance of the MMAS algorithm without and with local search methods. It can be observed from Figure 9.15 that when no local search is used the data is skewed to the left. This is also repeated in the case of using SW. The top whisker is much longer than the bottom whisker. In the case of using SI, DA, and BI the whiskers are absent. In the case of the IC method the length of the whiskers is shorter than the length of the box. As the problem size increased to 15 workstations, as shown in Figure 9.16, the results are very similar to the results obtained in Figure 9.15.

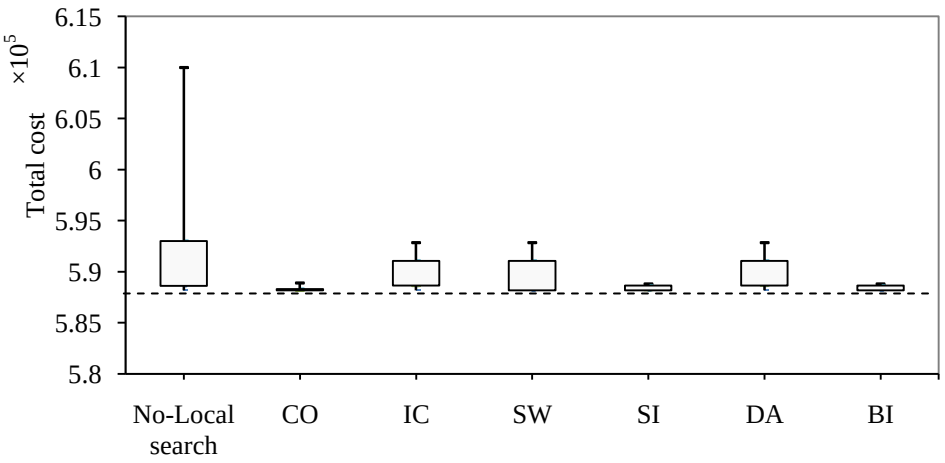


Figure 9.15: Box-plot showing the performance of MMAS with and without local search for 12 workstations problem

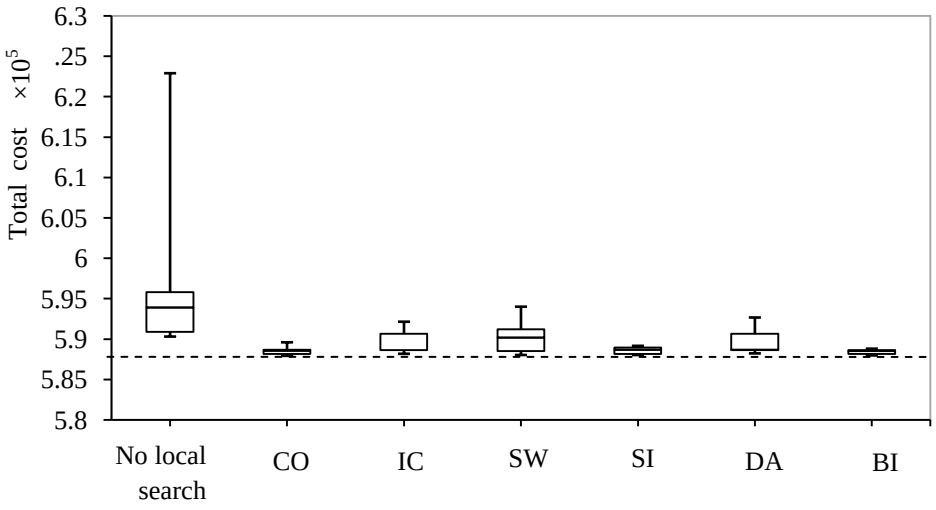


Figure 9.16: Performance of MMAS with and without local search for 15 workstations problem

The results determined that the performance of the MMAS significantly improved by using local search methods. It can be seen that the median of the MMAS algorithm with local search is closer to the optimal solution, which is indicated by a dashed line, than the median of the MMAS without using local search. This good performance can also be observed when the number of feasible solutions is significantly increased.

9.5 Stagnation measures

An indication of the performance of MMAS can be given by the development of the average branching factor during the algorithm's run. During the running of the MMAS algorithm, if the distribution of the pheromone on the trails becomes too unbalanced due to pheromone depositions the ants will generate very similar solutions, causing the search to stagnate. In order to enable the algorithm to detect such situations a branching factor is implemented in the AOIS problem to measure stagnation, which can indicate how explorative the search behaviour of the ants is. Using the average λ-branching factor as described in chapter 5, it is somehow possible to measure the size of the area under analysis at any moment. In general, the best solutions are found when the average branching factor is rather low, that is when the

algorithm has an almost converged space (Pellegrini and Moretti, 2009). In fact, the average branching factor at which the best trails are found also depends on the problem size, as shown in Figure 9.17. To increase the exploration of the MMAS, which is one of its features described in chapter 6, the pheromone trails are initialised to their upper trail limit if the average branching factor $\overline{\lambda}$ indicates that the MMAS is near convergence. It can be seen that as the number of iterations increases the solution moves closer to better results.

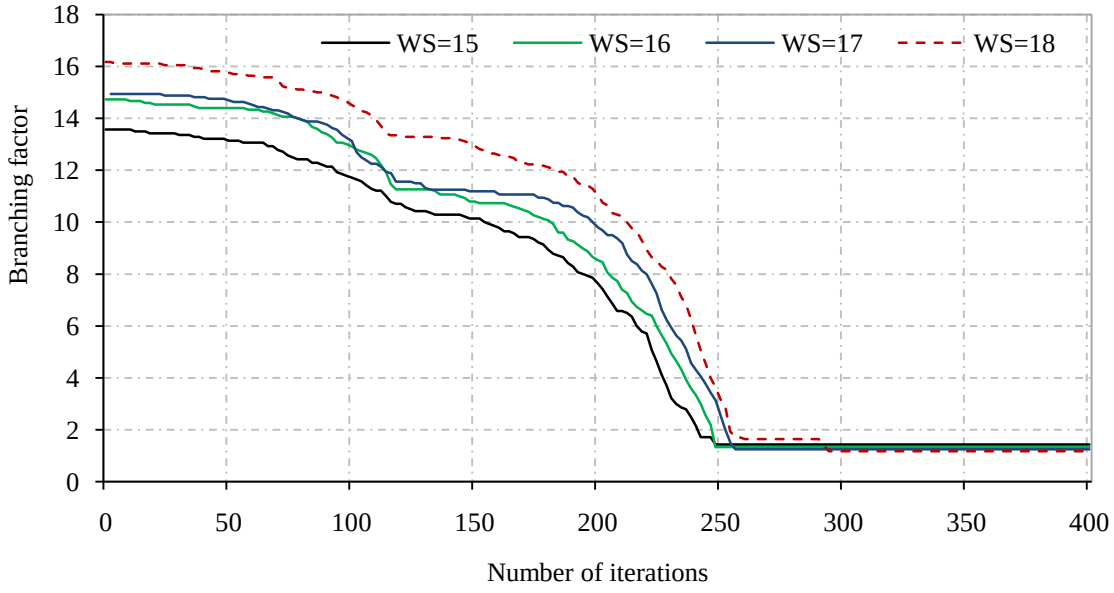


Figure 9.17: Plot of average branching factors for the AOIS problem

The branching decreases with the number of iterations. In the AOIS problem, the pheromone trails are initialised to their upper trail limit if $\overline{\lambda}$=1.4 in the case of n=15 workstations, $\overline{\lambda}$= 1.3 in the case of n=16 workstations, $\overline{\lambda}$=1.25 in the case of n=17 workstations and $\overline{\lambda}$=1.17 in the case of n=18 workstations, as shown in Figure 9.17. The experimental results confirmed that these average branching factors at which the best trails are found in the AOIS problem. Since $\overline{\lambda}$ does not change greatly from iteration to iteration, $\overline{\lambda}$ is measured every fifth iteration. These measures can be used as termination criteria or to reinitialise the algorithm

when no new areas of the search space are explored. In the AOIS problem these measures are used to increase the exploration of the search space as described in section 9.1.

9.6 Global and iteration best versus mixed strategy

As described earlier in this chapter, updating the pheromone trails for the MMAS algorithm was done using a mixed strategy s^{ib+gb}. The aim of this strategy is to obtain stronger exploration of the search space early in the search and stronger exploitation of the overall best solution later in the run. This mixed strategy specifies that in the first 300 iterations the iteration best ant s^{ib} is used to update the pheromone trails, and then every tenth iteration the global best ant s^{gb} is used for the pheromone trail update. To test the performance of this mixed strategy it was compared with the other two methods for updating pheromone trails, s^{ib} and s^{gb}. The parameter values for the MMAS algorithm were set as follows: $\alpha = 5$, $\beta = 1$ and $\rho = 0.02$. The rest of the parameters remained the same. The MMAS algorithm was applied on all 50 cases for the same case study (the general cost model case study).

The results are illustrated in Table 9.3. It can be seen that the mixed strategy used to update the pheromone trails showed better performance in terms of solution quality than the other two strategies. The box-plot in Figure 9.18 shows the performance of MMAS for the MMAS variants. The length of the whiskers far exceeds the length of the box. In all methods the top whisker is much longer than the bottom whisker. The line of MMAS-S^{gb} is gravitating towards the top of the box. In the other methods the line is close to the centre. The results confirmed that the performance of the MMAS when using the mixed strategy is much better than the other two strategies. It can be seen that the median of the mixed strategy is closer to the optimal solution, indicated by a dashed line, than the iteration best ant and the global best ant. In general, using the global best ant to update the pheromone trails does not seem to be a very good idea for MMAS. Nevertheless, the global best ant may sometimes be used to

reinforce the trials to direct the search more strongly. A main advantage of doing so is that a faster convergence of the algorithm can be achieved.

Table 9.3: Comparison between different strategies for updating pheromone trails

MMAS variants	Average number of optimal solutions (%)	Best average DFOS	Worst average DFOS
s^{ib}	76	0.00015	0.0186
s^{gb}	74	0.00020	0.0189
s^{ib+gb}	79	0.00012	0.0186

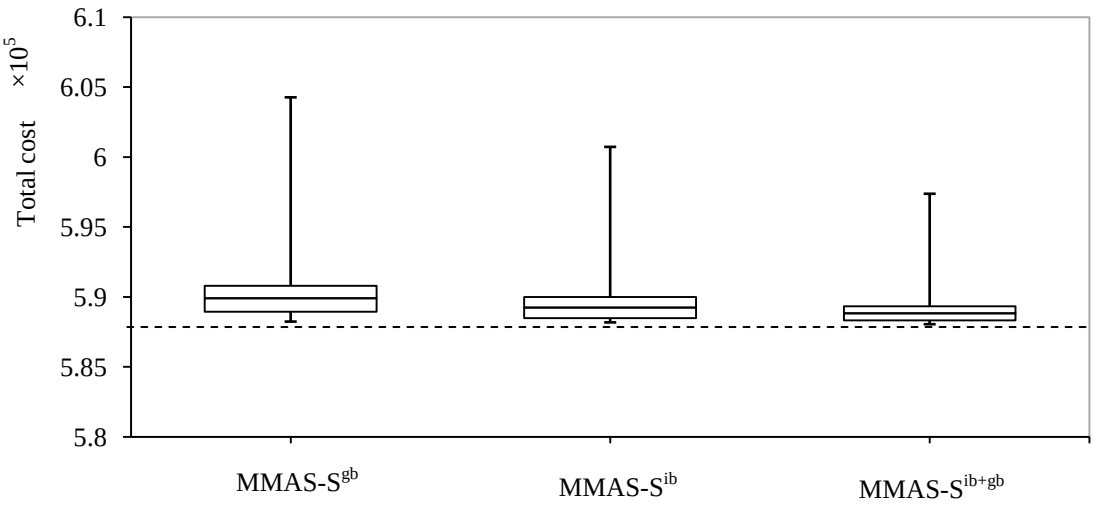


Figure 9.18: Performance of MMAS for different strategies for updating pheromone trails

9.7 Comparing variants of MMAS

Three different variants of the MMAS algorithm were developed and tested. In the first variant, denoted $MMAS_{10+ls}$, 10 ants were used and every ant applied a local search to its trail. In the second variant, $MMAS_{+ls+ib}$, the algorithm started with a fixed number of 10 ants and then the number of ants that applied the local search was successively increased by one after a certain number of iterations. The third variant used the MMAS algorithm without local search, denoted $MMAS_{-nls}$. Table 9.4 shows the results obtained by the different variants of the MMAS algorithm during the setting of the parameters.

Table 9.4: Comparison between different variants of the MMAS algorithm

MMAS variants	Best average DFOS	Worst average DFOS	Average processing time (seconds)
$MMAS_{10+ls}$	0.0001	0.018	16
$MMAS_{+ls+ib}$	0.0002	0.026	14
$MMAS_{-nls}$	0.005	0.026	13

The comparison between the three variants of the MMAS algorithm in terms of solution quality and processing time was based on the CEM. It can be seen that the $MMAS_{10+ls}$ showed better performance than the other two variants in terms of solution quality. It can also be seen that the best average DFOS obtained by the three variants were very close to the optimal solution. On the other hand, the $MMAS_{+ls+ib}$ algorithm needed less processing time to reach a near-optimal solution compared with the other two variants. It was found that by adapting the number of ants applying the local search, a good compromise between convergence speed and solution quality can be obtained. The computation results presented in this section suggest that, especially for larger problems, using the $MMAS_{+ls+ib}$ algorithm may be advantageous. In the variant $MMAS_{-nls}$ algorithm, the processing time was slightly higher compared with the other two variants. This is because the algorithm needed more iterations to arrive at a near-optimal solution.

9.8 Conclusion

In summary, the ratio between α and β is identified as the key deriver to the convergence speed using MMAS, and thus to finding the optimal solution. The results showed that the larger the ratio between α and β, the faster the convergence speed and thus the faster an optimal solution is found. In terms of sensitivity analysis, it was found that the parameters α, β and ρ are less sensitive within the specified range as the number of workstations is significantly increased. It was also found that using a mixed strategy to update the pheromone trails showed better performance than the other methods. The results showed that applying

the local search to the MMAS algorithm has significantly improved the performance of the algorithm. This good performance can even be observed when the number of feasible solutions increased significantly. It was observed that the MMAS algorithm, when combined with local search methods of crossover, single insertion and block insertion, present better performance than when combined with other local search methods. Furthermore, the experimental results suggest that it is important to use heuristic information with the MMAS algorithm in order to obtain a high quality solution. The computational results with respect to solution quality and computation times confirm that the effectiveness of the MMAS algorithm lies in its considerably shorter execution time and robustness. After tuning the parameters for the MMAS algorithm, the algorithm will be applied to four experiments to demonstrate its effectiveness. This is presented in the next chapter.

Chapter 10
Experimental results and discussion

The AOIS problem has been approached using a number of different techniques reported in this chapter that vary in their methods and efficiency. The complete enumeration method (CEM) is used as benchmark to test the MMAS algorithm against the rule of thumb (ROT), a pure random search algorithm (PRS), SA, GA and PSO algorithms. In total, four experiments were conducted to test performance of the MMAS algorithm against the other methods. These experiments utilised different experimental parameters to tackle the AOIS problem, such as size of the problem, assumptions, and methods used. The performance of the MMAS algorithm was compared to standard methods as well as those in the experimental case studies. The developed algorithms have been coded in MATLAB and executed on a 2.2 GHz CPU with 4 GB RAM in order to remain comparable in terms of the computational time.

10.1 Experiment 1

Experiment 1 uses the same general cost model case study used in Section 8.1 of chapter 8. The general cost model case study consisting of 12 processing workstations arranged in a serial manner in order to allocate five inspection stations. With this number of workstations, the full enumeration of the search space can be generated in a reasonable amount of time. The same experimental parameters for the developed general cost model and their ranges apply, as described in Section 8.1. The batch size is assumed to be 1000 and the sample size is set to be 80 for each case. These experimental parameters are used to randomly generate 100 different cases. These 100 cases are stored in database and will be reused in the other developed methods in this experiment as shown in Figure 10.1. Two performance measures

are used to evaluate the algorithms: deviations from the optimal solution (DFOS) and execution saving time. The DFOS is calculated by the following equation (10.1):

$$\text{DFOS} = \left(\frac{\text{Total cost of the algorithm}}{\text{Total cost of CEM}} - 1 \right)\% \qquad (10.1)$$

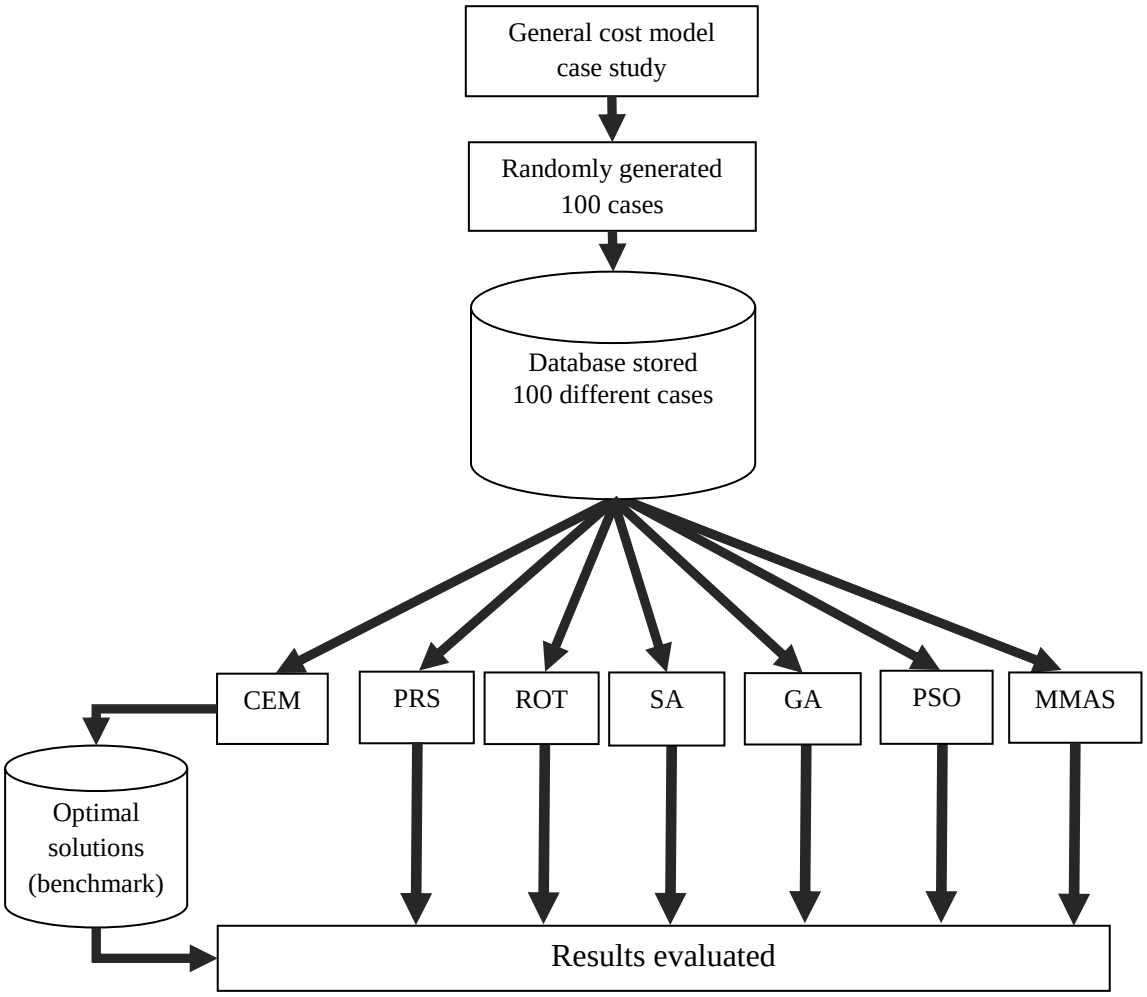


Figure 10.1: General cost model methodology experiment

Computer Execution Time (ET) is the CPU time it takes to execute the problem-solving computer programs. An efficient method should provide significant savings in terms of execution time over the CEM. The savings execution time is determined by equation (10.2):

$$\text{Execution time} = \left(1 - \frac{\text{ET of the algorithm}}{\text{ET of CEM}} \right)\% \qquad (10.2)$$

The AOIS problem will be tackled using six different algorithms. These are CEM, rule of thumb (ROT), SA, GA, PSO and MMAS. For each of the 100 test cases, 800 evaluations were performed by the algorithms and repeated 30 times to generate average for the test cases. The objective of these experiments is to find the allocation of a limited number of inspection stations in a serial multistage manufacturing process, with a reduction in the total cost as the end result. These methods will be presented in the next subsections.

10.1.1 Complete enumeration method

The CEM was applied to the general cost model case study with 100 different randomly generated cases. Here, the aim is to find optimum inspection plans that have the lowest total cost for each case. Each case study contains many inspection plans that are equivalent to the required number of inspection stations (a limited number of inspection stations) in this experiment. However, not all inspection stations are economically equivalent. Figure 10.2 displays the histogram of the optimal total cost for the studied cases produced by the CEM for the 100 cases.

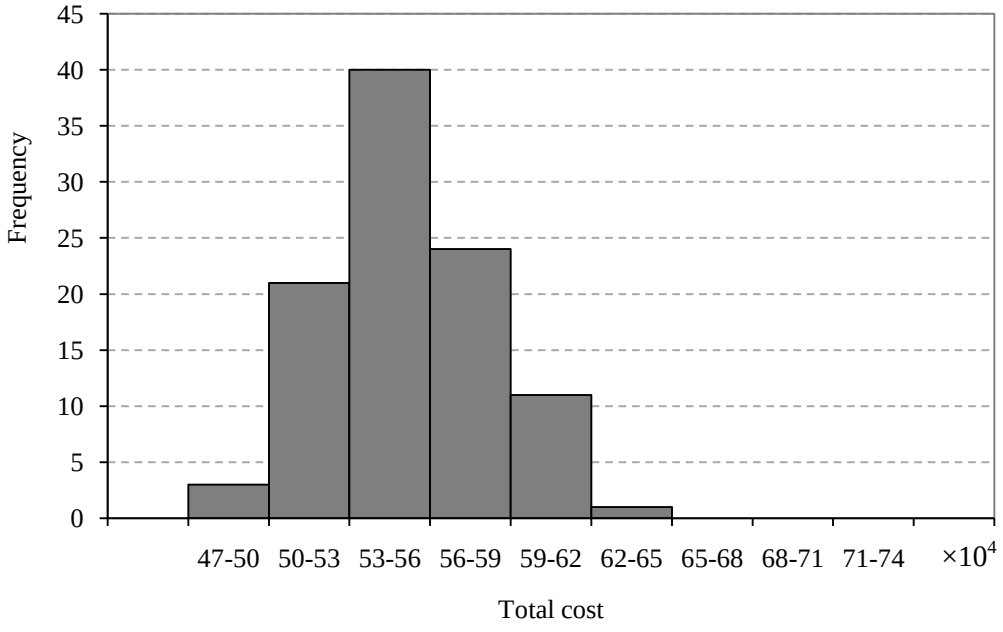


Figure 10.2: Histogram for the optimal cost from CEM

These total costs represent the lowest (optimal) costs of the inspection plans in each of the studied cases, and will be used as the benchmark for testing the other methods in this chapter. The testing will be carried out by measuring DFOS for each case study solved by the other methods. An effective method should provide solutions (total cost) very close to this optimal solution. The average total cost for the experiment was 550,063.6 with a standard deviation of 31,776.

10.1.2 Rule of thumb-a

The rule of thumb method (ROT-a) is based on the following two simple rules (Lee and Chen, 1996): (1) inspect *before* costly operations (U_k) in order to avoid the high processing costs for items that are already defective, and (2) inspect *after* operations that generally result in a high rate of defectives (Z_k) in order to avoid processing defective items in subsequent operations. The rule of thumb method was applied to the same 100 different random cases, the results which are illustrated in Figure 10.3. The histogram displays the distribution of the total cost produced by the ROT-a method. It can be seen that this histogram is also skewed to the right, with a peak between 530,000 and 680,000 and a long tail to 740,000. The average total cost for the studied cases with the ROT-a method was 605,222. The average similarity of the ROT-a method to the optimal average was 88.59%, indicating that many of inspection plans do not conform to the optimal cost. In addition, it was found that the standard deviation obtained by the ROT-a method for the studied cases was 45,631.

Figure 10.4 shows the DFOS percentage histogram for the studied cases achieved by the ROT-a method. The average DFOS was 0.0884, with a standard deviation of 0.054. The best and worst values of DFOS obtained by the ROT-a method were 0.0012 and 0.227, respectively.

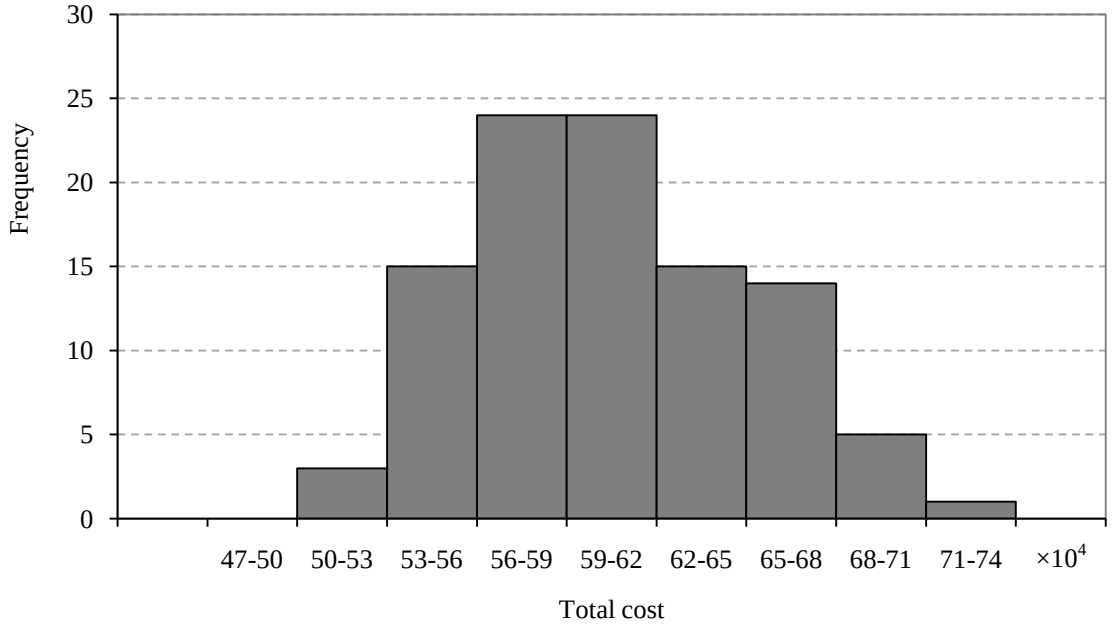


Figure 10.3: Total cost histogram for the ROT-a method

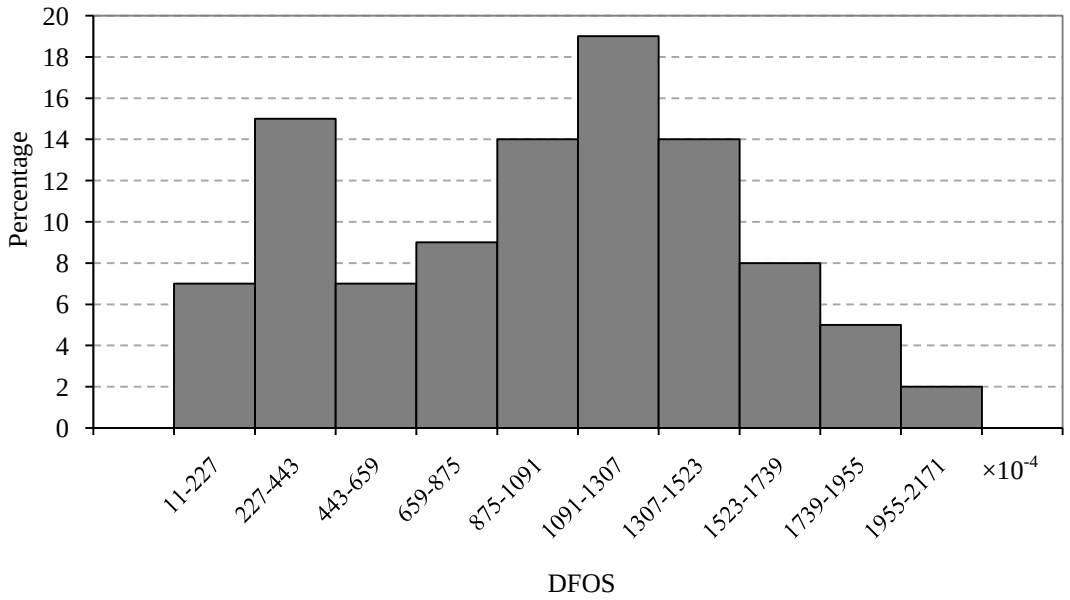


Figure 10.4: Percentage of DFOS histogram for the ROT-a method
(is not the same scale as the other methods)

10.1.3 Simulated annealing (SA)

Simulated annealing coupled with local search was implemented to solve the same 100 different cases. SA is a non-deterministic algorithm used to solve difficult optimisation

problems. It starts from an initial solution (possibly randomly generated) and generates a new solution in each step, which is either accepted or rejected according to the acceptance probability function as described in Section 4.3.1. The simulated annealing procedure and the pseudo-code are given in Appendix B. When implementing SA for the AOIS problem, the execution was terminated when either the optimal value is reached or if there is no improvement in the optimal value for a number of temperature reduction stages.

The results obtained by the SA method are illustrated in Figure 10.5. The histogram shows the distribution of the total cost obtained by the SA method. The average total cost for the experiment was 550,347.

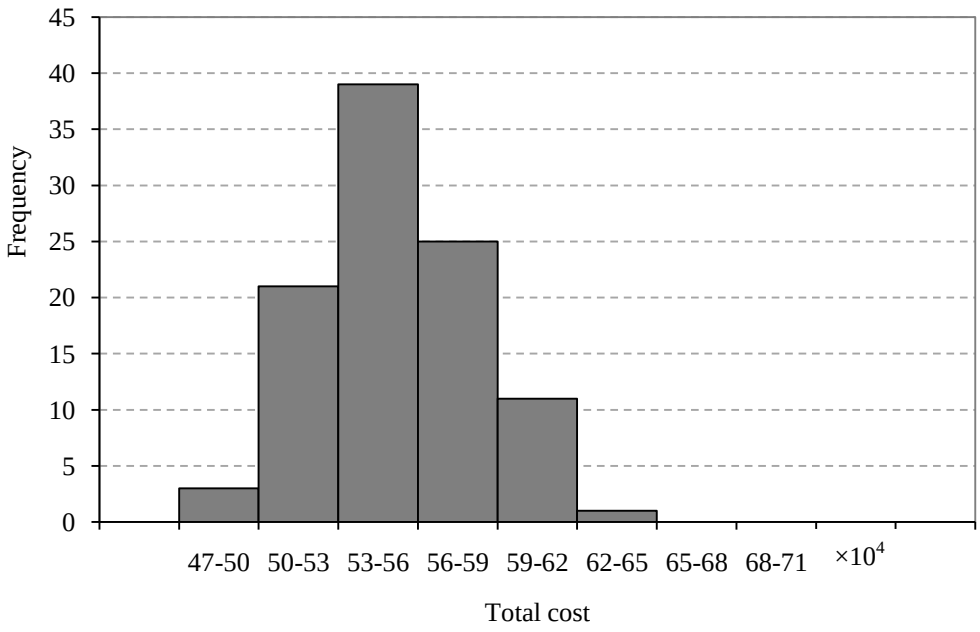


Figure 10.5: Histogram of the total cost from the SA method

The average similarity to the optimal average was found to be 99.94%, which indicates that most of the inspection plans are equivalent to the optimal solution. In addition, the standard deviation for the total cost for the SA method was 32,678 having a similarity to the optimal standard deviation of 99.92%. It can be seen in Figure 10.6 that SA reaches the optimal solution in 65% of the conducted number of experiments. The average DFOS for the studied

cases achieved by the SA method was 0.0006, with a standard deviation of 0.0012. Generally, the results of DFOS that were not optimal were nevertheless close to the optimal solution, with the best and the worst results being 0.0007 and 0.0080, respectively.

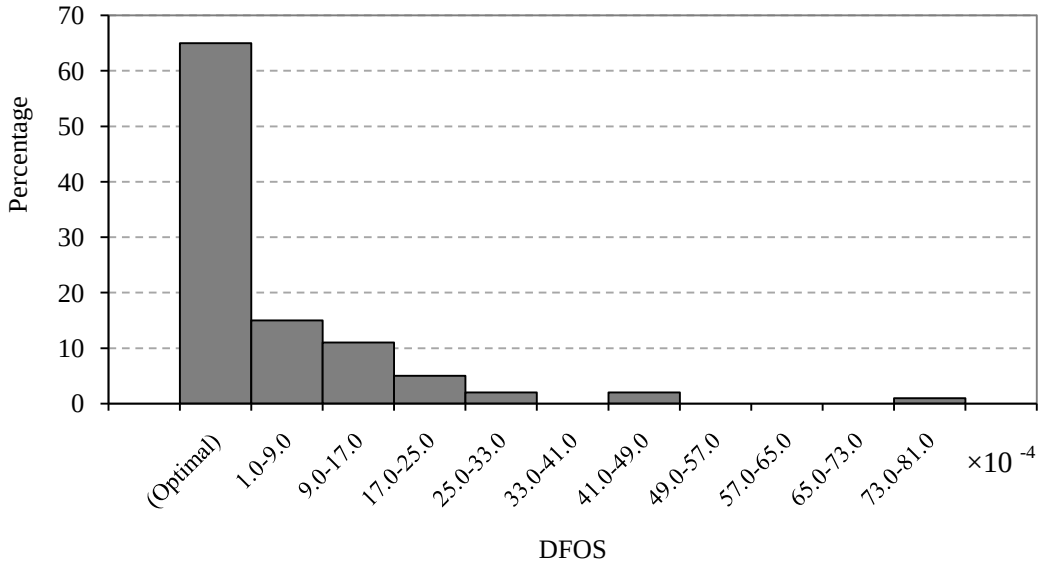


Figure 10.6: Percentage of DFOS histogram for the SA method

10.1.4 Genetic algorithm (GA)

The Genetic algorithm coupled with local search methods was also applied to the same 100 different cases for general cost model case study. GA is also a non-deterministic algorithm used to solve difficult optimisation problems. In the GA technique, a single initial solution, most likely randomly generated, initial solution is used, starting with a population that is representative of the search space. This is generally done by randomly generating a fixed number of individuals. The algorithm then applies several procedures to the population in an attempt to improve their fitness. GA is based on the idea that most solutions contain at least some useful information, and that by sharing it, a better solution can be constructed. It should be noted that the parameters for the GA are tuned in chapter 8. The GA procedure and pseudo-code are given in Appendix C. The results obtained by the GA method are illustrated in Figure 10.7. The average total cost of inspection plans for the same 100 random cases

obtained by the GA was 552,335, being 99.74% similar to the optimal average. In addition, the standard deviation was 32,874, with 99.30% agreement with the optimal standard deviation.

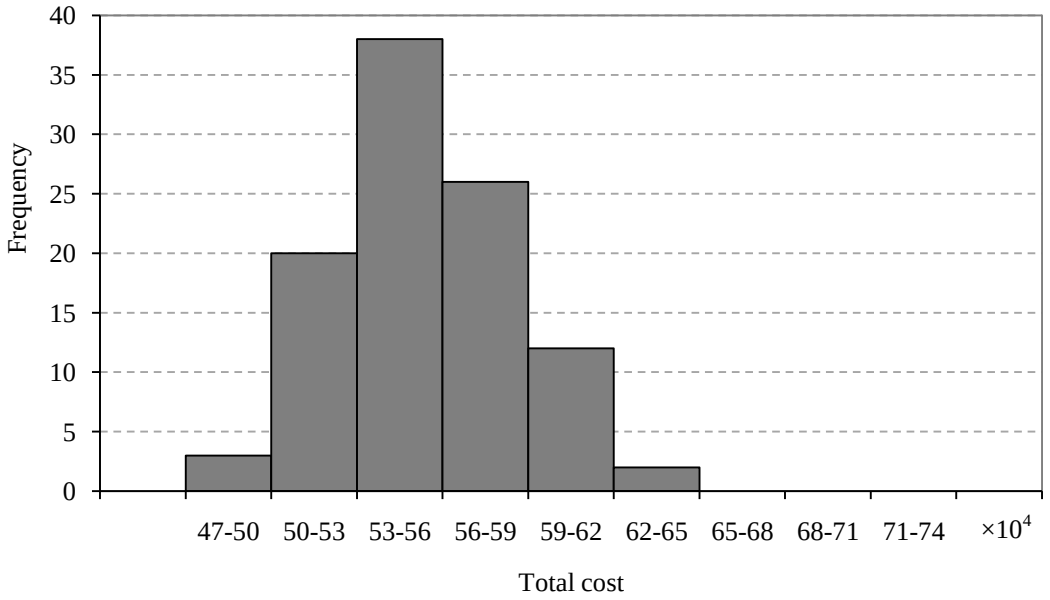


Figure 10.7: Histogram for the total cost from GA

Figure 10.8 displays the DFOS histogram achieved by GA for the experiment. As can be seen from Figure 10.8, GA reaches the optimal solution (identical to the CEM) in 25% of the conducted number of experiments. It was found that the average DFOS achieved by the GA was 0.0024, with a standard deviation of 0.002. The non-optimal results of the DFOS are as follows: a best result of 0.0007 and a worst result of 0.0094.

10.1.5 Particle swarm optimisation (PSO)

Particle swarm optimisation is a stochastic optimisation technique and also a population based search algorithm, inspired by social behaviour of bird flocking or fish schooling. After tuning the parameters for the PSO algorithm in chapter 8, the PSO algorithm is applied to the same general cost model case study. As in the preceding sections, the results obtained by the PSO algorithm are then tested against the optimal results obtained by the CEM. The results

obtained by PSO are illustrated in Figure 10.9. The average total cost achieved by the PSO algorithm was 550,350, having a more reliable performance with 99.94% agreement with the optimal average. In addition, the standard deviation of the total cost achieved by the PSO was 31,714.06.

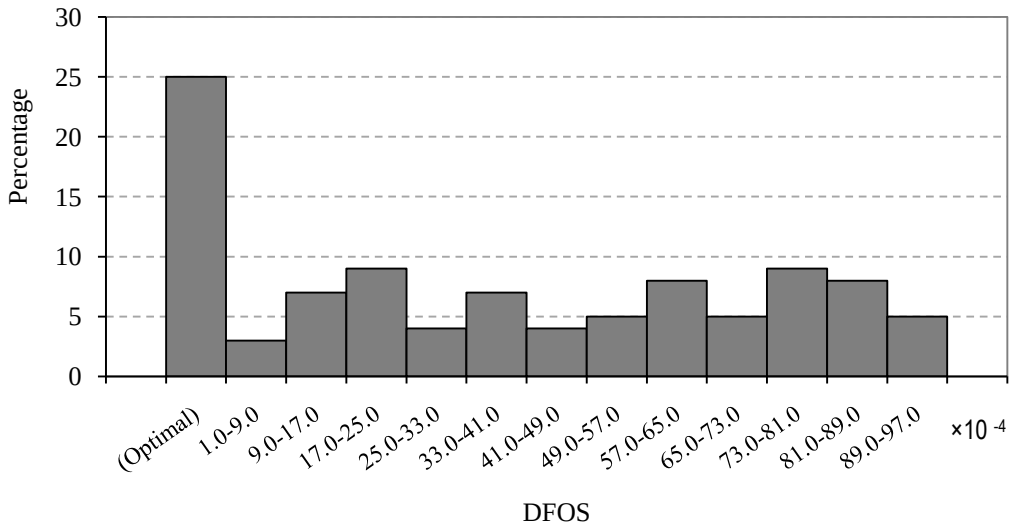


Figure 10.8: Percentage of DFOS histogram for GA

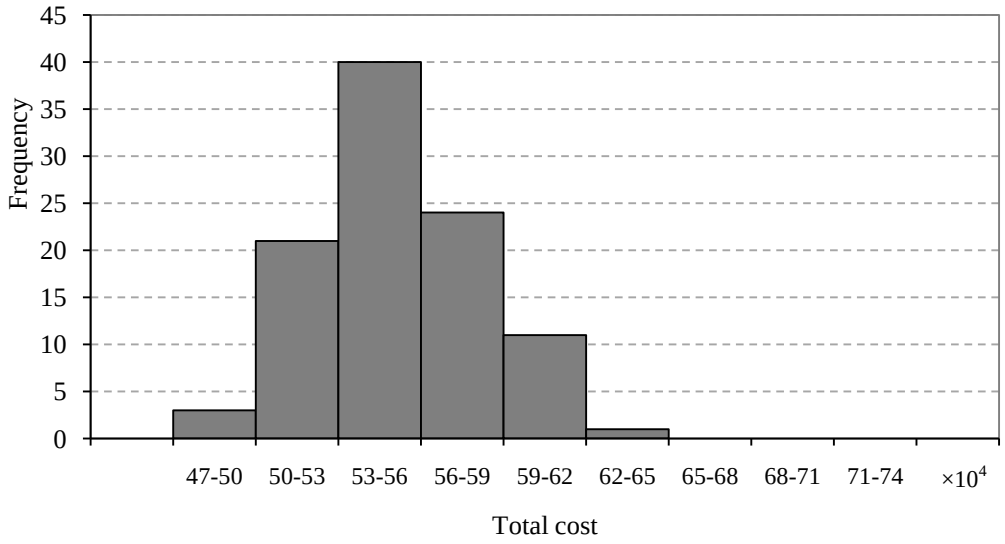


Figure 10.9: Histogram for the total cost from PSO

Figure 10.10 shows the DFOS percentage achieved by the PSO. It was found that the PSO algorithm reaches the optimal solution (identical to the CEM) in 65% of the conducted

experiments. In addition, it was found that the average DFOS for the studied cases achieved by the PSO algorithm was 0.0004, with a standard deviation of 0.00069. In general, the non-optimal results were close to the optimal solution, the best and worst DFOS being 0.00015 and 0.0028, respectively.

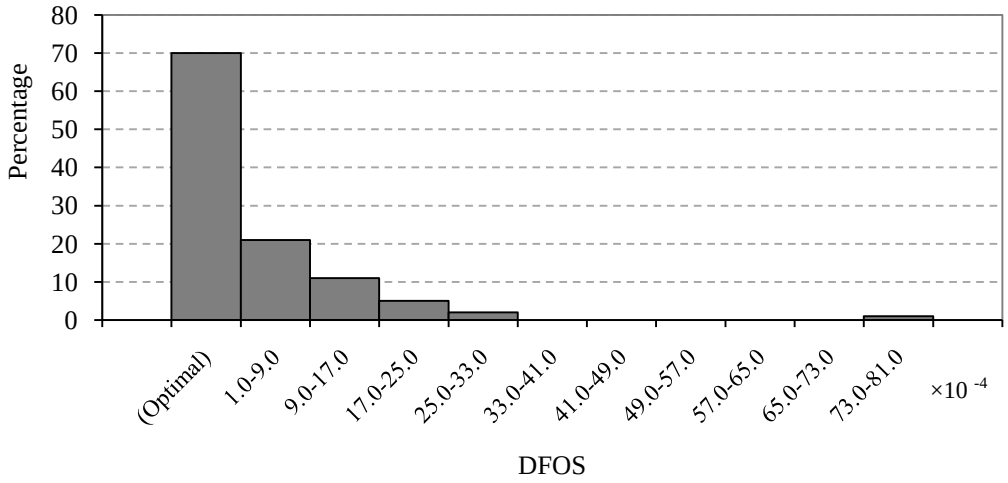


Figure 10.10: Percentage of DFOS histogram for PSO

10.1.6 Max-min ant system

After tuning the parameters for the MMAS algorithm in chapter 8, its performance is evaluated with the same general cost model case study. As in the previous sections, the results obtained by the MMAS algorithm are then evaluated against the optimal results obtained by the CEM. The following parameters were used by the MMAS algorithm: $\beta=1$, $\alpha=5$ and $\rho=0.02$. The results obtained by MMAS are illustrated in Figure 10.11. The average total cost achieved by the MMAS algorithm was 550,186, having a more reliable performance with 99.97% agreement with the optimal average. This high agreement also indicates that the vast majority of the total cost of inspection plans is very close to the optimum. In addition, the standard deviation of the total cost achieved by the MMAS was 31,586. Figure 10.12 shows the DFOS percentage achieved by the MMAS. It was found that the MMAS algorithm reaches the optimal solution (identical to the CEM) in the vast majority

of the conducted experiments 79%. In addition, it was found that the average DFOS for the studied cases achieved by the MMAS algorithm was 0.00017, with a standard deviation of 0.0005. These results are very small, indicating the effectiveness of the MMAS algorithm. In general, the non-optimal results were very close to the optimal solution, the best and worst DFOS results being 0.0001 and 0.0028, respectively.

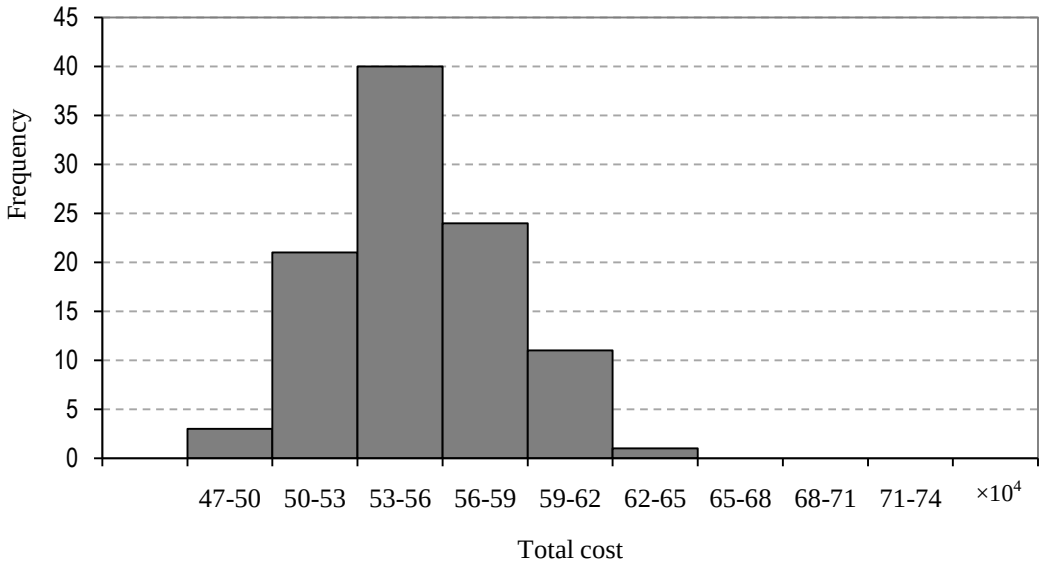


Figure 10.11: Histogram for the total cost from MMAS

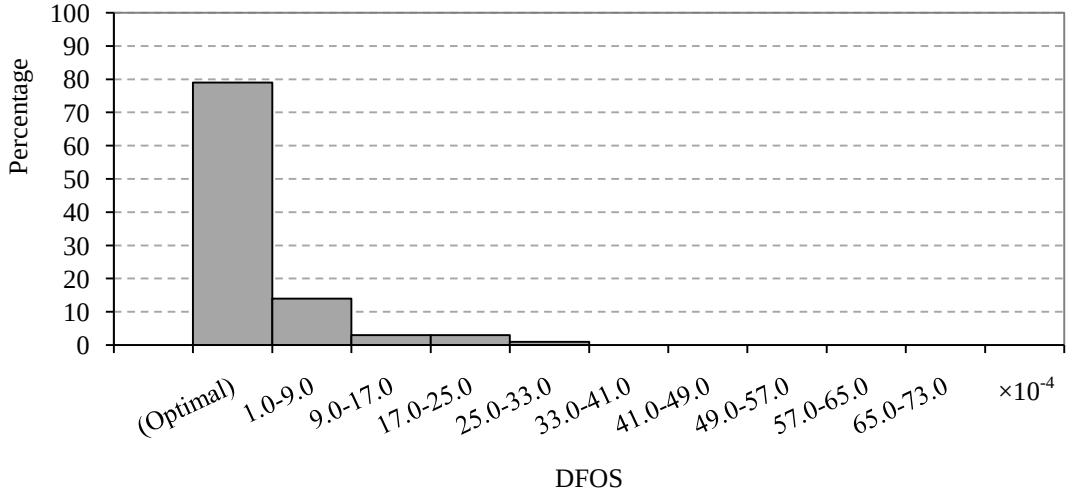


Figure 10.12: Percentage of DFOS histogram for MMAS

10.1.7 A pure random search (PRS)

A pure random search algorithm is developed to solve the AOIS problem. This is done to ensure that the solutions found by the MMAS algorithm are not just lucky strikes. The PRS algorithm is based on generating random solutions (10,000 random solutions) and the best a hundred solutions are selected. The results which are obtained by the PRS algorithm are illustrated in Figure 10.13. The histogram displays the distribution of the total cost produced by the PRS algorithm. The average total cost for the random solutions with the PRS algorithm was 628,819. In addition, it was found that the standard deviation obtained by the PRS algorithm was 61589.2

Figure 10.14 shows the DFOS percentage histogram for the studied cases achieved by the PRS algorithm. The average DFOS was 0.1244, with a standard deviation of 0.0560. The best and worst values of DFOS obtained by the PRS algorithm were 0.0303 and 0.2467, respectively.

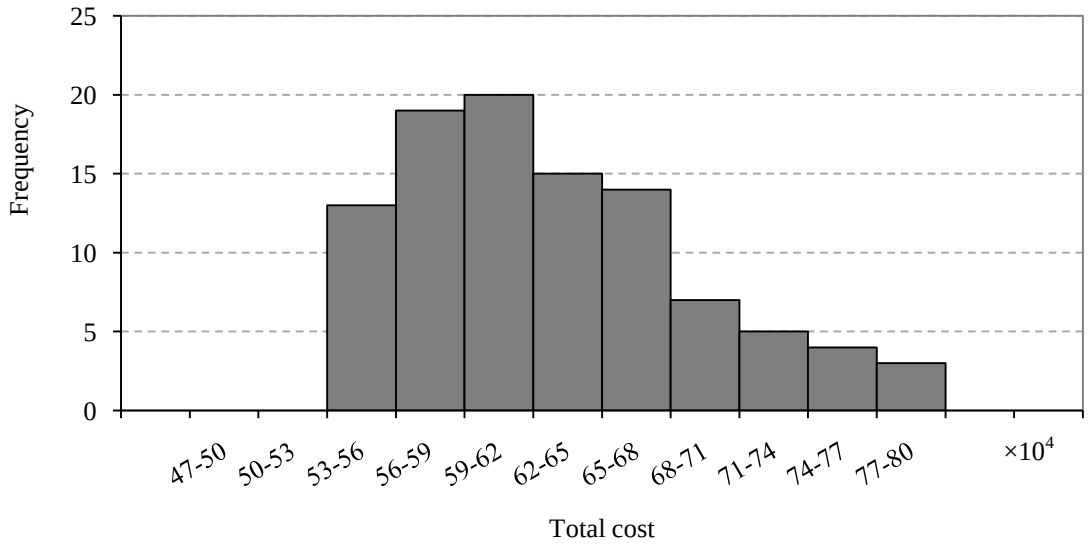


Figure 10.13: Total cost histogram for the PRS algorithm

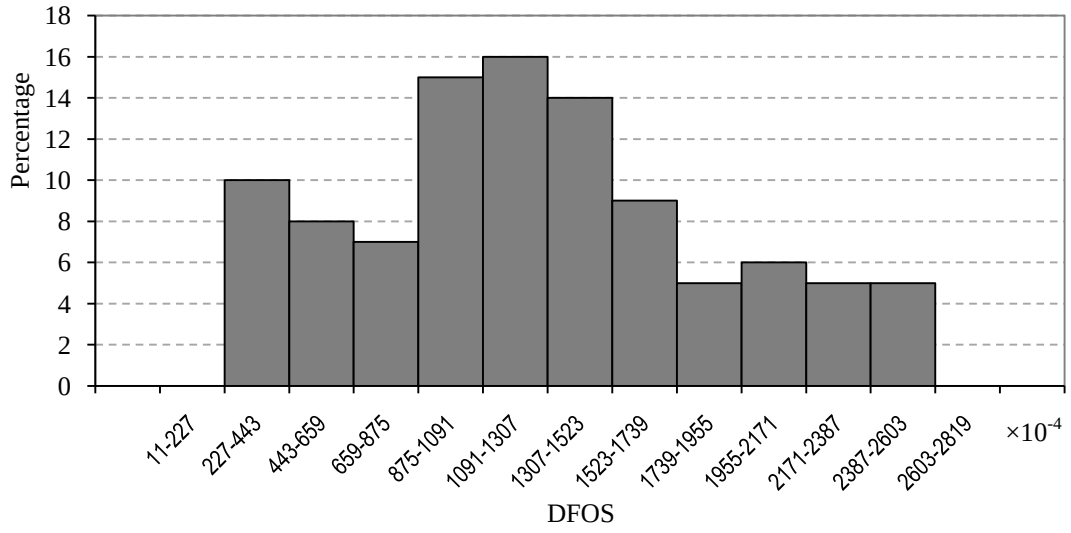


Figure 10.14: Percentage of DFOS histogram for the PRS algorithm
(is not the same scale as the other methods)

10.1.8 Comparing performance of the algorithms

Descriptive statistics are a quick and concise way to extract the important characteristics of a dataset. The main goal of descriptive statistics is to quickly describe the characteristics of the underlying distribution of a dataset through a simplified set of values. The most common technique for summarising data is the box-plot graph. Figure 10.15 illustrates box-plot for the performance of the algorithms approaching the AOIS problem against the CEM. It can be observed that the PRS and the ROT-a appear to have larger variability than the other four algorithms.

In addition both of them have long whiskers, which represent a wide ranging population. However, all algorithms are reasonably symmetric. Based on the lower and the upper whiskers there are no obvious outliers in any of the algorithm's data. As can be seen from the plot, the centre of the PRS and the ROT algorithm exceed the other algorithms. However, SA and GA algorithms appear to have similar centres, and all their centres are above the CEM

centre (optimal solution). Except for MMAS algorithm, the centre of the PSO algorithm is much better than the other algorithms.

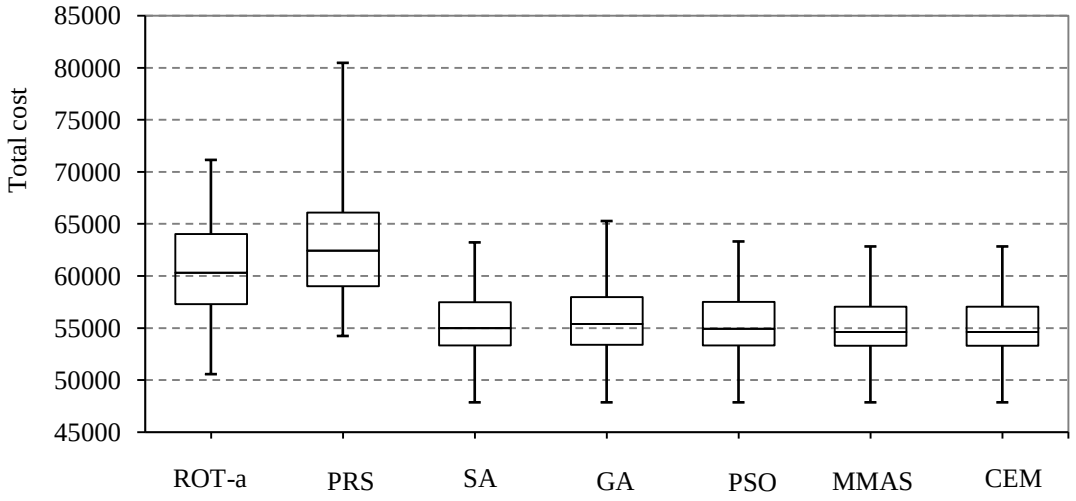


Figure 10.15: Performance of the algrithms against CEM for the AOIS problem

On the other hand, MMAS and CEM have similar centres. This confirmed that the vast majority of the inspection plans produced by the MMAS algorithm are identical to the CEM, which indicates the effectiveness of the MMAS algorithm. It is concluded that the performance of the MMAS algorithm in terms of solution quality is better than the other methods.

10.1.9 Solutions for experiment 1

The introduction of inspection stations into the production process, although constituting an additional cost; however, at some level of inspection points it is expected that such costs will be recovered from the benefits realized through the identification of defective items. Hence determining the optimal location of inspection stations is a very important decision. Of significance in such decisions are the trade-offs between the explicit costs of detection, repair and replacement associated with a particular inspection plan, and the implicit costs of unnecessary additional investment in a faulty item (and/or the costs of transmitting a nonconforming product beyond the boundaries of the process) when no inspection is

conducted. It is possible that not all locations of inspection stations are economically equivalent; more likely given the differentials in cost structures and process characteristics, some combinations of inspection places may prove to be economically preferable to others. The cost of AOIS problem involves many characteristics, including inspection errors (type I and type II), internal failure cost (rework and scrap), external failure cost (repair and replacement), inspection cost (fixed and variable) and manufacturing cost. As described in chapter 2, no literature considers all these characteristics together. However, the general cost model is developed to consider all these characteristics. As a result, the total manufacturing cost of a product can be reduced without affecting the quality of the product.

There are five inspection stations to be allocated in a serial multistage manufacturing process. The algorithms applied to the same general cost model case study. The total solutions produced by the algorithms are 600 solution, or inspection plans. It is impractical to discuss every individual solution. Therefore, in this section two cases (1.1 and 1.2) are selected in order to discuss their solutions, as shown in Tables 10.1 and 10.2. It should be noted that the studied cases are randomly generated, and therefore every case study has different characteristics for the workstations in terms of inspection cost, operation cost, rework cost and defect rates, as shown in Tables 10.2 and10.3. The inspection plans obtained by the SA, GA, PSO and MMAS algorithms are the inspection plans that have the lowest total cost.

Table 10.1: Inspection plans for case study 1.1

Algorithms	Workstations												Total cost
	1	2	3	4	5	6	7	8	9	10	11	12	
CEM	0	1	1	1	0	0	0	0	0	1	0	1	519,370
PRS	0	0	1	0	1	0	0	1	1	0	1	0	582,160
ROT-a	0	1	1	0	0	0	0	1	1	0	0	1	570,270
SA	0	1	1	1	0	0	0	0	0	1	0	1	519,370
GA	0	1	1	0	0	0	0	1	0	1	0	1	529,020
PSO	0	1	1	1	0	0	0	0	0	1	0	1	519,370
MMAS	0	1	1	1	0	0	0	0	0	1	0	1	519,370

Table 10.2: Inspection plans for case study 1.2

Algorithms	Workstations												Total cost
	1	2	3	4	5	6	7	8	9	10	11	12	
CEM	0	1	1	1	0	0	0	0	1	0	0	1	457,760
PRS	1	0	1	0	1	0	0	0	1	1	0	0	571,510
ROT-a	1	1	0	0	1	0	0	0	1	0	1	0	561,914
SA	0	1	0	1	1	0	1	0	0	0	1	0	461,840
GA	0	1	0	1	0	1	1	0	0	0	0	1	480,497
PSO	0	1	0	1	0	0	1	0	0	1	0	1	460,170
MMAS	0	1	1	1	0	0	0	0	1	0	0	1	457,760

In case study 1.1, the MMAS placed the inspection stations after workstations 2, 3, 4, 10 and 12. As can be seen from Table 10.3 the inspection stations at workstations 2,3 and 4 are placed before processing workstations 3, 4 and 5,which characteristically have a higher operating cost (U_3=99.797, U_4=97.723 and U_5^-86.601). As a result, to avoid processing items those are already defectives in subsequent processing workstations by continuing processing them. In addition, workstations 2, 3 and 4 have a characteristically low inspection cost compared to the other workstations (IC_2=40.300, IC_3=41.491 and IC_4=40.655). As there are a limited number of inspection stations, the last two inspection stations are placed at workstations 10 and 12. For workstation 10 the inspection station is placed before processing workstation 11 which has a high operation cost (U_{11}=90.512) to avoid processing defective items in subsequent processing workstations, note that workstation 10 has a low inspection cost (IC_{10}=40.65) compared to the other workstations.

For workstation 12 this entails the avoidance of penalty costs. In particular, to guarantee that no defective items reach the customer. As can be seen from Table 10.1, many inspection plans obtained by SA, PSO and MMAS algorithms are identical to the solution obtained by the CEM. On the other hand, the inspection plan obtained by GA is different to the optimal solution. The difference lies in the location of the inspection station at workstation 4, which is placed at workstation 8.

Table 10.3: Experimental parameters for case study 1.1

Experimental parameters	Workstations											
	1	2	3	4	5	6	7	8	9	10	11	12
Z_k	0.1191	0.1323	0.1596	0.1516	0.1144	0.1157	0.0913	0.1675	0.1696	0.1401	0.1691	0.1799
U_k	73.518	87.129	99.797	97.723	86.601	73.613	74.468	72.594	89.400	86.733	90.512	89.386
IC_m	48.765	40.300	41.491	40.655	47.655	42.160	48.371	44.841	44.773	40.651	43.653	44.567
α_m	0.0286	0.0177	0.0280	0.0210	0.0201	0.0113	0.0271	0.0122	0.0164	0.0138	0.0189	0.0134
β_m	0.0150	0.0188	0.0206	0.0120	0.0273	0.0294	0.0284	0.0294	0.0156	0.0220	0.0164	0.0261
u_k	31.231	109.877	98.251	97.385	30.739	46.572	57.99	46.56	52.043	86.348	60.156	112.931
g_k	76.822	49.7105	77.612	76.981	79.636	76.018	51.326	52.546	60.783	73.476	79.613	71.638
δ_k	0.0670	0.0835	0.0560	0.0830	0.0804	0.0724	0.0873	0.0516	0.0556	0.0565	0.0727	0.0762

Table 10.4: Experimental parameters for case study 1.2

Experimental parameters	Workstations											
	1	2	3	4	5	6	7	8	9	10	11	12
Z_k	0.1097	0.1766	0.1641	0.1496	0.1531	0.1252	0.0999	0.1310	0.1745	0.1490	0.1790	0.1500
U_k	61.018	97.766	95.336	62.388	53.555	68.249	65.734	70.165	54.719	89.750	61.323	74.937
IC_m	49.079	40.598	41.714	40.912	46.367	47.040	44.171	48.143	41.760	43.765	42.336	44.401
α_m	0.0172	0.0263	0.0183	0.0225	0.0177	0.0284	0.0114	0.0187	0.0188	0.0122	0.0133	0.0178
β_m	0.0118	0.0170	0.0150	0.0198	0.0149	0.0253	0.0284	0.0189	0.0187	0.0237	0.0226	0.0104
u_k	52.977	85.158	95.232	104.579	32.870	55.745	51.374	74.618	114.360	43.620	32.106	109.837
g_k	46.374	53.665	68.739	78.009	76.893	66.2411	74.972	61.324	66.369	72.864	66.329	68.178
δ_k	0.0654	0.0887	0.0805	0.0692	0.0814	0.0740	0.0660	0.0866	0.0852	0.0638	0.0796	0.0614

This location aims to avoid additional costs that may result from workstation 9 with a higher operation cost (U_9=89.400), whilst at the same time detecting defective items from workstation 8 with a higher defect rate (Z_8=0.1675). It should be noted that despite the operation cost at workstation 9 (U_9=89.400) being higher than the operation cost at workstation 5 (U_5=86.601), and the defect rate at workstation 8 (Z_8=0.1675) being higher than that at workstation 4 (Z_4=0.1516), the total cost of the inspection plan obtained by the GA of 529, 020 is higher than the total cost achieved by the MMAS algorithm or the optimal cost of 519,370.

The reason for this is that the early inspection station which was placed at workstation 4 in the inspection plan of the MMAS detected defective items before wasting additional resources by continuing to process them in subsequent workstations. In other words, the additional costs resulting from workstation 4 is much greater than the unnecessary costs resulting from workstation 8. This shows the effectiveness of the MMAS algorithm, with the advantage of using adaptive memory. The memory capability in the MMAS algorithm allows the algorithm to keep a record of the lowest cost of the inspection plans.

Except for the PRS algorithm, inspection plan obtained by the ROT-a method different from all other algorithms particularly in comparison with the MMAS algorithm. Inspection points are placed before the workstations that have higher operation costs or after workstations that have higher rates of defective items. Thus the first and the second inspection stations are placed before workstations 3 and 4 which have higher operation costs (U_3=99.797 U_4=97.723), as shown in Table 10.3. Consequently the third inspection station at workstation 8 is placed before workstation 9 with a higher operation cost (U_9=89.400). Whereas the fourth and the fifth inspection stations are placed after workstations 9 and 12 respectively, which generate higher defect rates (Z_9=0.1696 and Z_{12}=0.1799).The results of the ROT-a method indicate that the operation costs and the defect rate at a particular workstation do not

directly influence the inspection decision. On the other hand, it was found to be essentially dependent on the processing costs and defect rates at other workstations. Regarding the PRS algorithm, the total cost of the inspection plan obtained by the PRS algorithm is 582,160, which is higher than that obtained by other algorithms. This is because the PRS algorithm did not have any operator to guide the algorithm to good solutions.

In case study 1.2, the MMAS algorithm placed inspection stations at workstations 2, 3, 4, 9 and 12. As can be seen from Table 10.4, workstations 2, 3, 4 and 9 have a characteristically high defect rate (Z_2=0.1766, Z_3=0.1641, Z_4=0.1496 and Z_9=0.1745). In addition, inspection costs at these workstations is characteristically relatively low compared to the other workstations (IC_2=40.598, IC_3=41.714, IC_4=40.912 and IC_9=41.760). Placing these inspection stations at those workstations aim to identify defective items before passing them to the subsequent processing workstations. As a result, the total cost of the inspection plan can be minimised. The last inspection station is placed at workstation 12. This leads to avoiding external failure costs and to guarantee no defective items reach the customer. Regarding the inspection plan obtained by the PSO algorithm, it is slightly different from the optimal solution. The difference is that the inspection station placed in the optimal solution at workstation 3 is now placed at workstation 7. Furthermore, the inspection station at workstation 9 in the optimal solution is now placed at workstation 10. This combination of inspection stations led to a greater total cost compared with the inspection plan produced by the MMAS algorithm.

As can be seen from Table 10.4, the inspection plan achieved by the GA is different from the optimal solution. The difference is found in the placement of the inspection station at workstation 3 in the optimal solution, which is now placed at workstation 6. Also the inspection station at workstation 10 in the optimal solution is now placed at workstation 7.

The total cost obtained by this inspection plan is greater than the cost obtained of the MMAS algorithm.

Concerning the inspection strategy obtained by the SA algorithm, it is also different from the optimal solution. The difference lies in that the inspection station which in the optimal solutions placed at workstation 3, in this case is placed at workstation 5. A further difference is in the inspection station at workstation 9 in the optimal solution, which is placed at workstation 7 and in the inspection station at workstation 12 in the optimal solution which on this occasion is placed at workstation 11. As a result, the total cost obtained by SA algorithm is much greater than the total cost achieved by the MMAS algorithm.

Regarding the solutions obtained by ROT-a, the inspection plan is different from all other algorithms. Inspection points are placed before those workstations that have higher operation cost or after workstations that have a higher number of defectives items. The first, the second and the third inspection stations are placed before workstations 2, 3 and 10 which have characteristically higher operation costs (U_2=97.766, U_3=95.336 and U_{10}=89.750), as shown in Table 10.3. The fourth and the fifth inspection stations are placed after workstations 5 and 11 which have characteristically higher defect rates (Z_5=0.1531 and Z_{11}=0.1790). Concerning the inspection strategy obtained by the PRS algorithm, it was found that the total cost of the inspection plan obtained by the PRS is much greater than the total cost obtained by the other algorithms.

- **Summary**

The optimal inspection policy in a serial multistage manufacturing process has been studied. It has been proven that the solution approach using the MMAS algorithm gives better quality results in comparison with the other relevant algorithms when considering total inspection cost as the performance measure. The interpretation is that the MMAS algorithm outperforms the other algorithms because is enhanced with heuristic information. By using the heuristic

information the probable search space (the search space most likely to be explored) a much smaller space than the original search space will be explored. The purpose of the heuristic information in the MMAS algorithm is to guide the ants toward identifying promising regions of the search space. The search space in the general cost model case study for the AOIS problem is 2^{12}=4096 possible combinations for allocating inspection points. The heuristic information reduces this search space by guiding ants when assigning inspection stations to workstations with a high operation cost, defect rate and low inspection cost. Placing inspection stations at these workstations leads to reducing any unnecessary and avoidable costs, such as the cost of additional processing operations on a defective part in subsequent processing workstations. As a result, the total cost of the inspection plan can be minimised. In addition, the memory capability in the MMAS allows the algorithm to keep a record of previous search paths using the pheromone matrix. This matrix includes the paths (inspection positions) that the ants have visited. Thus, the path with the lowest cost will be used more frequently by subsequent ants. Furthermore, by applying a local search to the solutions that the ants have found the total cost can be further reduced. The local search works exchange inspection points, as a result of which the cost structure of the inspection plan is changed. This leads to improving the performance of the algorithm.

10.1.10 Discussion

The results of the MMAS algorithm and the other developed methods in Experiment 1 are summarised in Table 10.5. This experiment used a general cost model case study consisting of 12 processing workstations arranged in a serial manner in order to allocate five inspection stations. With this number of workstations, the full enumeration of the search space can be generated in a reasonable amount of time. This led to the use of CEM as a benchmark to compare the algorithms. The experimental parameters of the general cost model were used to generate 100 different cases, which represent the characteristics of different manufacturing

systems. For each of the 100 test cases, 800 evaluations were performed which are repeated 30 times to generate average for the test cases.

Table 10.5: Performance of developed methods for Experiment 1

Methods	% of optimal solution	Average total cost(£)	σ Total cost	Average DFOS	σ DFOS	Best* result (DFOS)	Worst result (DFOS)
CEM	100	550,063	31,776	(Datum)			
SA	65	550,347	32,689	0.0006	0.0012	0.0007	0.0080
GA	25	552,335	32,874	0.0024	0.002	0.0007	0.0094
PSO	70	550,350	31,714.06	0.0004	0.0006	0.00017	0.0028
MMAS	79	550,186	31,586	0.0001	0.0005	0.00012	0.0028
ROT-a	0.0	606,221	45,630.6	0.0994	0.0510	0.0011	0.2147
PRS	0.0	628,819	61,589.2	0.1244	0.0560	0.0303	0.2467

*Best result (DFOS): excluding the solutions that coincide with CEM optimal.

The results show that the MMAS algorithm clearly outperformed the other methods in all performance measures. It was found that the MMAS algorithm could find the optimal solution (identical to the CEM) in vast majority 79% of case studies. In addtion, MMAS was able to identify high-quality solutions with an average DFOS of 0.0001, which is clearly superior to the other methods. Likewise, it was found that the standard deviation obtained by the MMAS algorithm was much better than the other methods, also indicating the strength of the MMAS algorithm. In addition, the best and the worst results achieved by the MMAS algorithm are significantly better than the other methods, demonstrating the effectiveness of the MMAS algorithm. On the other hand, the average DFOS for the PSO method was better than the SA, and GA methods. In addition, the best and the worst results achieved by the PSO method is much better than the SA and GA methods. SA is based on only dealing with a single best solution, that having the best cost for one of the objectives, and not with a set of non-dominated solutions. Accepting worst solution enables the search to escape from the local optima, but this may also cause revisiting of previously evaluated points in the search

space and which also means the search is not adequately diversified. It was found that the average DFOS for the SA method was better than that for the GA method. In addition, the worst DFOS value 0.008 for SA was better than that of the GA method 0.0094; however, the best DFOS values for these two methods ended up being the same. Except for the PRS algorithm, the solution quality obtained by the ROT-a method was less than all the other methods in the comparison. Table 10.5 also shows that the solution quality obtained by the PRS algorithm was worse than all the other methods in the comparison. This was expected, because the PRS algorithm was based on placing inspection stations purely randomly and did not have any operator to guide the algorithm to good solutions.

Regarding running times, Table 10.6 shows the average execution time of the applied methods and the percentage of time saved in comparison with the CEM, by using Equation (10.2). With the exception of the ROT-a method, the MMAS algorithm shows a faster execution time than the other three methods. On the other hand, the percentage of time saved for the ROT-a method is the greatest among all the methods. The reason for this is due to the simplicity of the ROT-a computational process; it is completed in just one iteration. However, the solution quality produced by the ROT method is rather far from the optimal solution.

Table 10.6: Time performance of Experiment 1 methods compared to CEM

Method	Average execution time(seconds)	Time saved (%)
CEM	90	0.0
PRS	3	96
ROT-a	2	97
SA	45	50
GA	55	38
PSO	40	55
MMAS	35	61

10.2 Experiment 2

The data used in Experiment 2 are based on case study by Engin et al. (2008), which used real data that involved the manufacturing of an engine valve. Engin et al. (2008) proposed a fuzzy model solved with genetic algorithms for attribute control charts in multistage processes. The aim in the study by Engin et al. (2008) was to find acceptance numbers (e.g. determine whether to accept or reject a production lot of material) at each stage of the multistage process, resulting in the minimisation of cost at each stage. They applied their model to an engine valve manufacturing firm. The valves went through 24 or 36 different processing operations. Figure 10.16 is shown the operations of 24 processing workstations. These operations consist of different machines that are equipped with computer-controlled machinery. It should be noted that this case study was not originally an AOIS problem.

This case study was selected because it is considered a large problem comparing to the previous case study (general cost model) from which the performance of the MMAS algorithm against the SA, GA, PSO, PRS and ROT-a methods could be evaluated. In addition, this case study is based on real data and most of the required data for the MMAS algorithm are available. The number of feasible solutions in this experiment is 2^{24} =16,777,216.

It AOIS problem, as the size of the problem grows, so the number of inspection station allocation possibilities increases exponentially, CEM has computational times that are too long for practical purposes. Therefore, an optimal solution for this experiment cannot be obtained with CEM in a reasonable time. For this reason, the comparison between the algorithms will be based on absolute values rather than DFOS. Table 10.7 shows the experimental parameters and their ranges for Experiment 2. Because the case study of Engin et al. (2008) did not consider inspection errors which might occur at inspection points, the

developed general cost model for the AOIS problem in this experiment thus assumes that the inspection is performed free of error. This assumption is simply adapted by setting the inspection error parameters in the general cost model to: $\alpha=0$ and $\beta=1$.

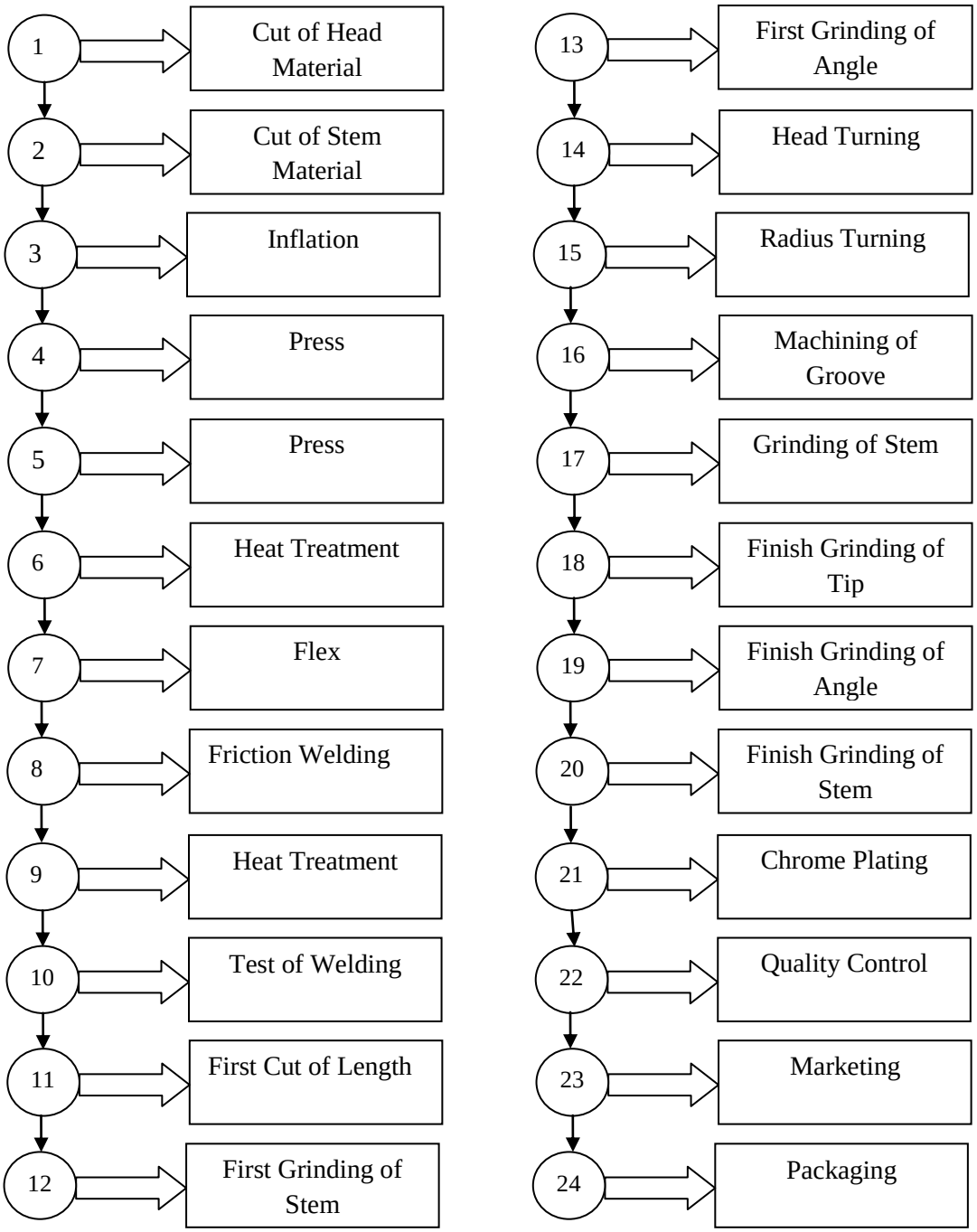


Figure 10.16: Operations of engine valve (Engin et al., 2008)

Table 10.7: Experimental parameters for Experiment 2

Parameters	Range	Brief description
B	1000	Batch size
U_k	[0.075, 0.085]	Unit manufacturing cost ($)
IC_m	[0.0075, 0.0085]	Unit inspection cost ($) (assumed)
Z_k	[0.004, 0.01]	Defective rate
u_k	[0.25, 0.270]	Unit scrap cost ($)
g_k	[0.018, 0.022]	Unit reworking cost ($) (assumed)
δ_k	[0.03, 0.09]	Repairing probability
Penalty cost	[2, 10]	Each defective item that reaches the customers ($) (assumed)

Because both the unit inspection and unit reworking costs are unavailable, they are assumed to be reasonable values. In reality, inspection cost is significantly less than manufacturing cost. Therefore, the inspection and reworking costs are assumed to be 10% and 25% of the manufacturing cost, respectively. Some previous studies used this assumption (e.g. Lee and Unnikrishnan, 1998). Penalty cost is the cost associated with the final production of undetected non-conforming items that reach the customer. Penalty cost is assumed in these ranges based on the price of the cost valves. The firm generally produces the valve at a cost of $1–$5 (Engin et al., 2008). This range of the penalty cost is expected to cover issues such as replacement, repairs, return products, and other services.

All developed methods (i.e., ROT-a, PRS, SA, GA, PSO and MMAS) were applied to 100 different randomly generated cases for solving the AOIS problem. These cases, generated by a uniform random number generator, represent the characteristics of different manufacturing systems. It is assumed that there are a limited number of inspection stations 8 available to be allocated to the different processing workstations; this number was chosen to match the value used by Engin et al., (2008). In terms of solution quality, it is important to report the average solution quality created by the applied methods, because for increasing problem sizes, this is the most important discriminating factor. Therefore, the evaluation of the performance of the

developed methods will be based on the average of the total cost and the average execution time. The results obtained by the methods in Experiment 2 are presented in Table 10.8. It can be seen that the MMAS algorithm performs considerably better than the other methods in terms of solution quality (average total cost). In addition, the standard deviation of the total cost obtained by the MMAS algorithm is less than the other developed methods, indicating a more reliable performance for the MMAS algorithm. These experimental results confirm that the MMAS algorithm outperforms the other methods even when the AOIS problem is significantly increased. Table 10.8 also shows that the solution quality produced by the PRS algorithm is much less than the other methods.

Table 10.8: Performance of the studied methods for Experiment 2

Methods	Average total cost ($)	σ Total cost	Average execution time (seconds)
PRS	2571.68	8.25	4
ROT-a	2556.65	10.08	3
SA	2505.56	4.09	90
GA	2508.44	4.95	110
PSO	2503.63	4.23	89
MMAS	2492.76	1.56	86

Regarding processing time, the MMAS has the least average processing time as compared to the other methods, except for the ROT-a method. Indeed, the average processing time provided by the ROT-a method is the smallest among all methods, due to its simplicity and the fact that it does not need more than one pass to arrive close to the optimal solution. In reality, the production management of firms are more interested in solution quality rather than processing time, as long as the execution time of the method is reasonable. The GA method has the highest average processing time because its steps to reach the solution are longer than the other methods, and it also incorporates a local search method. It should be noted that genetic algorithms require large number of response (fitness) function evaluations

depending on the number of individuals and the number of generations. Therefore, genetic algorithms may take long time to evaluate the individuals.

As described in section 5.5.2, the fitness distance correlation (FDC) for the AOIS problem indicates a strong correlation between the solution quality and the distance to the optimum. The high fitness distance correlation indicates that the search space of the AOIS problem is a globally convex, single-funnel landscape or big valley structure. A big valley structure means that local optima tend to be relatively close to each other and to the global optimum. In a big valley structure the MMAS can potentially drive the search towards an optimal solution or near optimal solution. In other words, this high correlation suggests that the local optima are radially distributed in the problem space, with the global optima as the centre, and the more distant the local optima are from the centre the worse their objective function values. Hence, by tracing local optima step by step, moving from one optimum to nearby slightly better ones, one can eventually reach a near global optimal solution.

10.2.1 Solutions for experiment 2

As in the previous section, the solutions obtained by the relevant algorithms for the engine valve case study are presented. Since it is impractical to discuss every individual solution obtained by the relevant algorithms, two cases (2.1 and 2.2) are selected in order to discuss their solutions. Tables 10.9 and 10.10 show the results and the experimental parameters respectively for case study 2.1. Tables 10.11 and 10.12 show the results and the experimental parameters respectively for case study 2.2. The inspection plans obtained by SA, GA, PSO and MMAS algorithms are the inspection plans that have the lowest total cost. In case study 2.1, no inspection is selected for the first workstation (cut off head material) for the inspection plan obtained by the MMAS method. This means that the cost avoidance of detecting defective products in workstation 1 does not outweigh the cost of performing an inspection at this workstation.

Table 10.9: Inspection plans Case study 2.1

Algorithms	Workstations																							Total cost
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
PRS	0	1	0	1	0	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	2670.8
ROT-a	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	1	2605.3
SA	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1	0	0	1	2510.3
GA	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	2515.6
PSO	0	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	1	2493.8
MMAS	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	0	0	1	2488.7

Table 10.10: Experimental parameters for case study 2.1

Experimental parameters	Workstations											
	1	2	3	4	5	6	7	8	9	10	11	12
Z_k	0.0056	0.0091	0.0097	0.0083	0.0056	0.005	0.0073	0.0073	0.0093	0.0097	0.0065	0.0083
U_k	0.0740	0.0817	0.0854	0.0796	0.0838	0.0821	0.0765	0.0770	0.0769	0.0846	0.0804	0.0824
IC_m	0.0082	0.0081	0.0080	0.0079	0.0084	0.0079	0.0082	0.0078	0.0080	0.0081	0.0075	0.0082
u_k	0.2075	0.2537	0.2065	0.2010	0.2132	0.2635	0.2607	0.2570	0.2193	0.2477	0.2067	0.248
g_k	0.0180	0.0216	0.0192	0.0203	0.0212	0.0207	0.0204	0.0198	0.0202	0.0216	0.0210	0.0199
δ_k	0.0832	0.0399	0.0499	0.0470	0.0863	0.0300	0.0465	0.0420	0.0797	0.0442	0.0738	0.0389

Table 10.10: Experimental parameters for case study 2.1 (continued)

Experimental parameters	Workstations											
	13	14	15	16	17	18	19	20	21	22	23	24
Z_k	0.005	0.0095	0.0089	0.0082	0.0043	0.0080	0.0097	0.0098	0.0047	0.0049	0.0078	0.0095
U_k	0.0810	0.0757	0.0834	0.0826	0.0752	0.0840	0.084	0.0792	0.0782	0.0780	0.0834	0.0802
IC_m	0.0078	0.0075	0.0084	0.0076	0.0081	0.0077	0.0078	0.0078	0.0079	0.0084	0.0085	0.0077
u_k	0.2615	0.2540	0.2241	0.2166	0.2023	0.2423	0.2618	0.2473	0.2504	0.2549	0.2023	0.2050
g_k	0.0192	0.0196	0.0215	0.0181	0.0206	0.0201	0.0191	0.0212	0.0211	0.0193	0.0203	0.0193
δ_k	0.0498	0.0492	0.0465	0.0696	0.0537	0.0498	0.0890	0.0365	0.0489	0.0823	0.0721	0.0684

Table 10.11: Inspection plans for Case study 2.2

Algorithms	Workstations																							Total cost
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
PRS	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	2660.7
ROT-a	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0	0	1	2595.3
SA	0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	0	1	0	0	1	2508.4
GA	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0	0	1	2513.3
PSO	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	2503.33
MMAS	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	2490.5

Table 10.12: Experimental parameters for case study 2.2

Experimental parameters	Workstations											
	1	2	3	4	5	6	7	8	9	10	11	12
Z_k	0.0057	0.0044	0.0076	0.0071	0.0058	0.0068	0.0084	0.0050	0.0046	0.0057	0.00920	0.0082
U_k	0.0788	0.0791	0.0843	0.0781	0.0753	0.0796	0.0752	0.0826	0.0794	0.0775	0.0797	0.0785
IC_m	0.0076	0.0082	0.0083	0.0080	0.0078	0.0077	0.0076	0.0077	0.0080	0.0081	0.0075	0.0078
u_k	0.2662	0.2699	0.2536	0.2511	0.2679	0.2599	0.2648	0.2546	0.2565	0.2586	0.2689	0.2614
g_k	0.0181	0.0187	0.0213	0.0190	0.0208	0.0196	0.0212	0.0210	0.0184	0.0209	0.0219	0.0190
δ_k	0.0410	0.0336	0.0771	0.0890	0.0570	0.0340	0.0733	0.0344	0.0501	0.0678	0.0327	0.0468

Table 10.12: Experimental parameters for case study 2.2 (continued)

Experimental parameters	Workstations											
	13	14	15	16	17	18	19	20	21	22	23	24
Z_k	0.0092	0.0094	0.0062	0.0094	0.0058	0.0082	0.0096	0.0090	0.0074	0.0075	0.0089	0.0097
U_k	0.0780	0.0761	0.0766	0.0843	0.0807	0.0844	0.0845	0.0797	0.0796	0.0844	0.0791	0.0772
IC_m	0.0079	0.0079	0.0078	0.0076	0.0080	0.0078	0.0078	0.0083	0.0077	0.0080	0.0077	0.0083
u_k	0.2660	0.2553	0.2601	0.2685	0.2699	0.2602	0.2555	0.2502	0.2543	0.2636	0.2689	0.2635
g_k	0.0220	0.0180	0.0203	0.0213	0.0193	0.0183	0.0210	0.0196	0.0203	0.0210	0.0202	0.0187
δ_k	0.0331	0.0424	0.0730	0.0851	0.0894	0.0481	0.0540	0.0511	0.0337	0.0678	0.0689	0.0433

In addition, as can be seen in Table 10.10, the operation cost and defect rate at the first workstation are lower than in the other workstations. The first inspection station in the inspection plan created by the MMAS is placed before processing workstation 3 (inflation process) with a high operation cost (U_3=0.0854), and after workstation 2 (cut off stem material), which generates a relatively high defect rate (Z_2=0.0091) compared to the other workstations. This leads to avoiding processing items that are already defective in subsequent processing workstations. The second inspection station is placed after workstation 3, which has a high defect rate (Z_3=0.0097). This inspection station aims to detect defective items before allowing them to pass to successive workstations. The third, fourth and fifth inspection stations are placed before workstations 12 (first grinding of stem), 16 (machining of groove) and 18 (finishing grinding of tip) with high operation costs (U_{12}=0.0824, U_{16}=0.0826 and U_{18}=0.0840). These inspection stations aim to avoid unnecessary processing costs in subsequent processing workstations. The sixth, seventh and eighth inspection stations are placed at workstations 14 (head turning), 19 (finish grinding of angle) and 24 (packaging) which generate high defect rates (Z_{14}=0.0095, Z_{19}=0.0095 and Z_{24}=0.0097).

In contrast, these workstations have characteristically low inspection costs (IC_{14}= 0.0075, IC_{19}=0.0078 and IC_{24}=0.0077).The inspection station at workstation 24 aims to detect any defective items before reaching the customer. The inspection plan obtained by the PSO is different from the inspection plan obtained by the MMAS algorithm. The difference is in the location of the inspection station at workstation 14 (head turning) in the MMAS inspection plan which is placed at workstation 4 (press) in the PSO inspection plan. This cost structure of the PSO inspection plan led to a total cost higher than the total cost from the MMAS algorithm. The inspection plans obtained by GA, SA, ROT and PRS algorithms are also different from the inspection plan achieved by the MMAS algorithm. As a result, these

inspection plans led to higher total costs compared to the total cost obtained by the MMAS algorithm.

In case study 2.2, as can be observed in Table 10.11, the inspection plans obtained by the algorithms are different from the one in case study 2.1. This is because the characteristics of case study 2.2 in terms of operation cost, defect rate, rework cost and inspection cost are different from those in case study 2.1. The MMAS algorithm distributed the eight inspection stations through the serial line as can be seen in Table 10.12. Note that workstations 3, 8, 19 and 22 have characteristically high operation costs (U_3=0.0843, U_8=0.0826, U_{19}=0.0845 and U_{22}=0.0844). The first four inspection stations are placed at workstations 2 (cut off stem material), 7 (flex), 18 (finish grinding of tip) and 21 (chrome plating). This leads to avoiding unnecessary operation costs in subsequent processing workstations. Also from Table 10.12 it can be observed that workstations 11 (first cut off length) and 16 (machining of groove) have characteristically low inspection costs and high defect rate (IC_{11}= 0.0075, IC_{16}= 0.0076, Z_{11}=0.00920 and Z_{16}=0.0094). Therefore, the fifth and the sixth inspection stations are placed at these workstations. These inspection stations aim to detect the defective items and to avoid additional costs. Workstations 19 (finish of grinding angle) and 24 (packaging) feature high defect rates (Z_{19}= 0.0096 and Z_{24}=0.0097) compared with the other workstations, thus the seventh and the eighth inspection stations are located at these workstations.

In contrast these workstations characteristic with low inspection cost (IC_{14}= 0.0075, IC_{19}=0.0078 and IC_{24}=0.0077).The inspection station at workstation 24 aims to detect any defective items before reach to the customer. The inspection plan obtained by the PSO is different from the inspection plan obtained by the MMAS algorithm. The difference was in location of the inspection station at workstation 14 (head turning) in MMAS inspection plan is placed at workstation 4 (press) in PSO inspection plan. This cost structure of PSO inspection plan led to total cost higher than the total cost in the MMAS algorithm. The

inspection plans obtained by GA, SA and ROT-a algorithms are also different from the inspection plan achieved by the MMAS algorithm. As a result, these inspection plans led to high total cost comparing with the total cost obtained by the MMAS algorithm.

10.2.2 The trade-off between the MMAS and CEM

In this section, the trade-off between the CEM and MMAS algorithms is applied to the engine valves case study. As described in section 10.3, the average total cost of the inspection plan for the engine valves case study consisting of 24 workstations obtained by the MMAS was $2492.76. It is impractical to solve the problem using CEM in reasonable time due to computational complexity. Assuming that the average DFOS for the MMAS is 0.0029, this is the worst DFOS for the general cost model case study. The estimated optimal solution (CEM) for the 24 workstations is:

(average total cost of MMAS) – (DFOS ×average total cost of MMAS)

($2492.76) – (0.0029×$2492.76) =$2485.53.

The difference in average total cost between the MMAS and the estimated optimal solution per batch is:

(average total cost of MMAS) – (the estimated optimal solution): ($2492.76) – ($2485.53) =$7.23.

This is the cost of using MMAS per batch. As described in chapter 3, the duration of the computation time for the 24 workstations using CEM is about 2 days. The firm uses a batch of 1000 units and the average valve price is $3. Assuming that the company works one shift per day, the total production sold in one day is therefore: 1000×$3=$3000.

The waiting time for the optimal solution using CEM is about 2 days and will cost the company $6000. On the other hand, the additional cost of using the MMAS to obtain a near optimal solution for 365 days: $7.23×365=$2638.95.

It can be seen that the additional cost of using the MMAS algorithm of $2638.95 is much less than the waiting time cost of $6000. This additional cost of using the MMAS algorithm is compared with other additional costs of using GA. The GA is selected because is a very common algorithm in the literature. The average total cost obtained by the GA was $2508.44. The difference in average total cost between the GA and the CEM per batch is: (average total cost of GA) – (average total cost of CEM)

($2508.44) – ($2485.53) =$22.91. The difference in additional cost per batch between the MMAS and GA is: ($22.91) – ($7.23) =$15.68. The saving in cost of using MMAS over GA for 365 days: $15.68×365=$5723.

It can be seen that the MMAS algorithm saves $5723 annually over GA for the engine valves case study consisting of 24 workstations. It is interesting to know the difference between the best and the worst solutions obtained by the MMAS and PRS algorithm respectively for the engine valves case study consisting of 24 workstations. The difference in additional cost per batch between the MMAS and PRS is:

(average total cost of PRS) - (average total cost of MMAS)

($2571.68) – ($2492.76) =$78.92

$78.92×365=$28,805

The MMAS algorithm saving $28,805 annually over the PRS algorithm for the engine valves case study consisting of 24 workstations. The saving in cost of using the MMAS algorithm is also compared for the engine valve system consists of 36 workstations. Due to computational complexity CEM cannot be applied to this number of workstations. As a result, the saving in cost of using the MMAS algorithm is compared against GA for this number of workstations. The average total cost resulting from the MMAS algorithm and GA are $5276.6 and GA $5293.7, respectively. The difference in average total cost between MMAS and GA is: (average total cost of MMAS) – (average total cost of GA)

($5293.7) – ($5276.6) =$17. The saving in cost of using the MMAS algorithm over GA for 365 days is: $17×365=$6205

It can be seen that the MMAS algorithm saves $6205 annually over GA for the engine valves case study consisting of 36 workstations. In reality, the production management of firms is in the search for any potential cost savings that can keep the company in a good competitive position. It is evident that despite the CEM producing the optimal solution, in terms of the economic aspect is impractical to allocate inspection places using CEM as the number of workstations (WS) increases (e.g. WS $\geq$ 24). This impracticality in CEM has led to the use of the MMAS algorithm that sacrifices the guarantee of finding the optimal solution in order to find a satisfactory solution in a reasonable time. This agrees with what the vast majority of case studies in the literature review pointed out, that CEM is an impractical way of finding the optimal solution as the number of workstations increases significantly. It is concluded that using MMAS leads to a saving in money by minimising the computation time, the total cost of the product and keeps the company in a good competitive position.

10.3 Experiment 3

Experiment 3 is based on the case study of Rau and Chu (2005), which considered some experimental heuristic rules-of-thumb in a serial production system. This experiment is different from the previous experiments and it is selected for the following reasons: (i) it uses different experimental parameters from the previous experiments; (ii) it uses 100% inspection of items when an inspection station is located after a workstation; and (iii) it uses heuristic method, so the MMAS algorithm can be tested with different method from the previous methods.

Rau and Chu (2005) developed a heuristic method (HM) that allocates inspection stations based on the manufacturing cost and the probability of non-conformity at each processing

workstation. The procedure of the HM works iteratively until the total profit of the objective function cannot be further improved. In Rau and Chu (2005), the HM was implemented on a serial production system with various sizes of workstations, with an aim to find the optimal location of the inspection stations. Table 10.13 presents the parameters that were used in the case study. Rau and Chu (2005) used the average and standard deviation of DFOS to measure the solution quality of their HM. However, Rau and Chu (2005) did not specify the number of cases generated for each workstation size.

Table 10.13: Experimental parameters for Experiment 3

Parameters	Range	Description
B	1000	Batch size
U_k	[80,150]	Unit manufacturing cost (£)
IC_m	[2, 4]	Unit inspection cost (£)
Z_k	[0.01, 0.03]	Defective rate
α_m	[0.001, 0.002]	Type-I error
β_m	[0.001, 0.002]	Type-II error
u_k	[10, 15]	Unit scrapping cost (£)
g_k	[10, 15]	Unit reworking cost (£)
δ_k	[0.2, 0.3]	Repairing probability (assumed)

Therefore, it was decided that 50 cases should be randomly generated using a uniform random number generator for each workstation size in order to calculate the DFOS average and standard deviation. Rau and Chu (2005) assumed 100 % inspection to items processed in a workstation if an inspection station is scheduled after it in the sequence. This assumption is simply adapted in the MMAS by considering full inspection at each inspection station. MMAS is applied to these cases for each workstation size, and the results obtained by the MMAS algorithm are presented in Table 10.14. It should be noted that the HM results were obtained by Rau and Chu (2005), in which a complete enumeration method was used as a benchmark to test their HM. It can be seen that the MMAS algorithm outperforms the HM in terms of DFOS average and standard deviation.

Table 10.14: Performance of the methods in Experiment 3

Number of workstations	HM		MMAS	
	Average DFOS	Average σ DFOS	Average DFOS	Average σ DFOS
6	0.000	0.000	0.000	0.000
8	0.002	0.001	0.000	0.000
10	0.003	0.005	0.000	0.000
12	0.0001	0.001	0.000	0.000
14	0.003	0.002	0.000	0.000
16	0.006	0.007	0.0002	0.0007

As can be seen from Table 10.14, MMAS achieves the optimal solution in all test cases when the number of workstations is less than 16. When the number of workstations is equal to 16, the MMAS algorithm produces a solution very close to the optimal solution (i.e., a DFOS average of only 0.00024). As the problem size increases, the solution quality obtained by the HM becomes much less than that of the MMAS algorithm. It is concluded that the MMAS algorithm performs significantly better than the HM in terms of solution quality.

10.3.1 Solutions of experiment 3

As described above in the previous sections it is impractical to discuss every individual solution obtained by the MMAS algorithm (50 cases), thus one case study (3.1) is selected in order to discuss their solution. This solution is represented the case study for 16 workstations. Tables 10.15 and 10.16 show the solutions and experimental parameters respectively for case study 3.1. It should be noted that Rau and Chu (2005) did not describe the solutions obtained by the HM. Table 10.16 presents only the solutions obtained by the MMAS algorithm and CEM for case study 3.1. In case study 3.1, the MMAS placed the five inspection stations after workstations 1, 3, 7 10 and 16.

As can be seen from Table 10.16 inspection stations at workstations 1,3 and 7 are placed before processing workstations 2, 4 and 8 which characteristically have a higher operating cost (U_2=144.10, U_4=147.16 and U_8=145.37).

Table 10.15: Inspection plans for case study 3.1

Algorithms	Workstations																Total cost
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
CEM	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	161,780
MMAS	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	161,742

0: no inspection and 1: full inspection

Table 10.16: Experimental parameters for case study 3.1

Experimental parameters	Workstations															
	1	2	3	4	5	6	7	8	9	10	11	12	3	14	15	16
U_k	109.52	144.10	135.45	147.16	125.90	82.49	139.43	145.37	127.51	133.04	132.01	107.45	125.88	91.983	129.42	82.22
IC_m	2.05	2.22	2.19	2.16	3.38	2.63	2.23	2.06	2.87	2.76	3.53	3.59	2.37	2.97	2.89	3.29
Zk	0.025	0.022	0.015	0.023	0.023	0.013	0.012	0.020	0.029	0.016	0.021	0.014	0.025	0.015	0.021	0.024
α_m	0.0019	0.002	0.0015	0.0011	0.0011	0.0013	0.0018	0.0013	0.0018	0.0012	0.0019	0.0013	0.0012	0.0013	0.0016	0.0015
β_m	0.0015	0.0011	0.0011	0.0012	0.0019	0.0013	0.0013	0.0018	0.0013	0.0018	0.0012	0.0015	0.0011	0.0011	0.0013	0.0015
u_k	11.75	14.15	12.92	12.74	14.58	11.42	11.42	13.78	13.76	13.76	11.90	12.81	12.92	12.74	14.58	12.92
g_k	14.15	12.92	12.74	14.58	11.42	11.98	12.81	12.92	12.74	11.90	14.15	12.92	12.74	14.58	11.42	14.65
δ_k	0.259	0.246	0.201	0.233	0.216	0.279	0.231	0.252	0.216	0.260	0.226	0.265	0.268	0.274	0.245	0.279

As a result, to avoid processing items those are already defectives in subsequent processing workstations by continuing processing them. In addition, workstations 1, 3 and 7 which characteristically have a lower inspection cost comparing to the other workstations (IC_1=2.05, IC_3=2.19 and IC_7=2.23). Because there is limited number of inspection stations, the last two inspection stations are placed after workstations 9 and 16. Workstation 9 which characteristically has a higher defective rate (Z_9=0.029) to avoid processing defective items in subsequent processing workstations. For workstation 16 this entails avoidance of penalty costs.

10.4 Experiment 4

Experiment 4 is based on the case study of Shiau (2007), which considered a serial multistage manufacturing system. Shiau (2007) developed a unit cost model to represent the overall performance of an advanced manufacturing system. Since the problem gets more complex as the problem size increases, Shiau (2007) therefore developed a genetic algorithm to allocate a limited number of inspection stations in a serial multistage manufacturing system, with the aim of reducing the total manufacturing cost. The experimental parameters used in the experiment are presented in Table 10.17. Fifty cases were randomly generated by Shiau (2007) to represent the characteristics of different manufacturing systems. Shiau (2007) applied a GA method to these 50 cases, and its performance was measured in comparison to the CEM optimal solution. Because the defective rate, repairing probability, and inspection errors were not specified by Shiau (2007), these have been based on ranges from similar values in previous literature (Lee and Unnikrishnan, 1998). In addition, some external failure cost items are missing; thus, external failure cost items are represented as an aggregated penalty cost based on similar values from previous literature (Raz and Kaspi, 1991). However, it should be noted that the penalty cost usually depends on the type and complexity of the product produced by the company.

Table 10.17: Experimental parameters for Experiment 4

Parameters	Range	Description
B	1000	Batch size
U_k	[100, 250]	Unit manufacturing cost(£)
Z_k	[0.01, 0.06]	Defective rate (assumed)
IC_m	[1, 10]	Unit inspection cost (£)
α_m	[0.01, 0.03]	Type-I error (assumed)
β_m	[0.01,0.03]	Type-II (assumed)
u_k	[50, 150]	Unit scrapping cost (£)
g_k	[50, 100]	Unit reworking cost (£)
δ_k	[0.2, 0.3]	Repairing probability (assumed)
Penalty cost	[500, 1600]	Each defective item that reaches the customers

The MMAS algorithm is applied to the 50 cases for each workstation size, which were randomly generated using the same experimental parameters. Table 10.18 shows the results from Shiau (2007), in which the case study was tested for a range of feasible solutions. This number of feasible solutions is equivalent to 5–13 workstations. However, it was not clear how many workstations were used in the calculation of the average DFOS for the GA in Shiau (2007), because he was interested in investigating the computational execution time for these numbers of workstations. Therefore, the MMAS algorithm tested for the largest number of workstations, 13. Table 10.18 shows the results obtained by the MMAS algorithm compared to those in Shiau (2007). It was found that MMAS achieved the optimal solution in all test cases, allowing us to conclude that the MMAS algorithm outperforms the GA method in terms of solution quality.

Table 10.18: Performance of the methods in Experiment 4

Number of workstations	Average DFOS	
	GA	MMAS
13	0.0041	0.000

10.5 Rule of thumb-b

After conducting the four experiments, it is interesting to analyse the behaviour of the MMAS algorithm for allocating inspection stations. Therefore, the results (inspection plans) obtained by the MMAS algorithm for 100 case studies are extensively investigated. These results represent the lowest total cost of inspection plans obtained by the MMAS algorithm. This leads us to develop a good new rule of thumb. The developed rule of thumb will be very useful for industry, as no tuning parameters are needed. As discussed in section 10.1, these cases were randomly generated in order to represent the varying characteristics of different manufacturing systems. These characteristics including inspection errors (type I and type II), internal failure cost (rework and scrap), external failure cost (repair and replacement), inspection cost (fixed and variable), defective rates and manufacturing cost. As described in section 6.2.2, some of these characteristics have a greater effect on the total cost of the product than others. The detailed investigation of the results is presented in Appendix G. The analysis of the solutions leads to the following observations:

1. The MMAS algorithm always locates an inspection station at the last workstation to guarantee that no defective items reach the customer.
2. The MMAS algorithm locates an inspection station after the machining operation with the highest probability of generating nonconforming parts, in order to avoid further work on units that should be scrapped. If two or more workstations have the same nonconforming parts, the MMAS algorithm locates an inspection station for the earlier workstation.
3. The MMAS algorithm locates an inspection station before the most costly machining operation, in order to avoid the high cost of the machining operations on parts that are already nonconforming. If two or more workstations have the same operation cost, the MMAS algorithm locates an inspection station for the earlier workstation.

4. The MMAS algorithm is biased towards locating inspection stations before costly machining operations rather than after workstations with the highest probability of generating nonconforming parts. Specifically, it was found that about 57-60% of inspection stations for inspection plans are placed before the most costly machining operation.

These observations are used to develop a new rule of thumb which is denoted by ROT-b. ROT-b is applied to the same engine valves case study consisting of 24 workstations (Engin et al., 2008) to solve the AOIS problem. The case study assumed that there are a limited number of eight inspection stations to be allocated to the different processing workstations. Based on these rules, the inspection stations are distributed through the processing workstations as follows:

1. An inspection station is placed at last workstation in order to avoid a penalty cost.
2. Four inspection stations (four inspection stations out of seven = 60 %) are placed before the most costly machining operations in order to avoid the high cost of the machining operation on parts that are already nonconforming. If two or more workstations have the same operation cost, the priority is to locate an inspection station before the earlier workstation.
3. The rest of the inspection stations (three inspection stations) are placed after the machining operations with the highest probability of generating nonconforming parts, in order to avoid further work on units that should be scrapped. If two or more workstations have the same nonconforming units, the priority is to locate an inspection station for the earlier workstation.

The performance of ROT-b is compared against ROT-a and the MMAS algorithm. To do so, ROT-b is applied to the same 100 cases. The aim is to calculate the average total cost and standard deviation. Table 10.19 shows performance of the applied methods. It can be seen

that the performance of ROT-b is much better than that of ROT-a, but is worse than the MMAS algorithm. Table 10.20 presents the solutions obtained by ROT-a and ROT-b for one case study selected from the 100 cases. The inspection stations in Table 10.20 are located based on the rules described by each of ROT-a and ROT-b. Table 10.21 shows the characteristics of each workstation in the selected case study, in terms of unit operation cost and defective rates.

Table 10.19: Comparing performance of ROT-a, ROT-b and the MMAS

Methods	Average total cost ($)	σ Total cost
ROT-a	2556.65	35.08
ROT-b	2526.51	32.26
MMAS	2492.76	1.56

It can be seen from Tables 10.20 and 10.21 that there is a strong link between the places of inspection stations and their characteristics. As an example, an inspection station is located at the last workstation. This is done to guarantee that no defective items reach the customer. Also, four inspection stations, which represent 60% of the remaining seven inspection stations, are placed before workstations 12, 15, 16 and 20 which have higher operation costs (U_{12}=.0.0832, U_{15}=00845,U_{16}=0.0845 and U_{20}=0.0827). The rest of the inspection stations are placed after workstations 6, 18 and 22 which are characterised by a high defective rate (Z_6=0.0083, Z_{18}=0.0084 and Z_{22}=0.0085). Although workstation 20 has the same defective rate (Z_{20}=0.0083) as workstation 6, according to ROT-b the priority is given to the earlier workstation which is workstation 6.

It is concluded that the analysis of behaviour of the MMAS algorithm for allocating inspection stations has led to develop a new a rule of thumb. The ROT-b performed much better than ROT-a. The advantage of the rule of thumb is its simplicity of method: no tuning parameters are needed, so it is preferred by industry.

Table 10.20: Inspection plans for the selected Case Study

Algorithms	Workstations																								Total cost
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
ROT-a	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0	2595.3
ROT-b	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	1	2503.33
MMAS	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	2490.5

Table 10.21: Unit operation cost and defective rates for the selected Case Study

Experimental parameters	Workstations											
	1	2	3	4	5	6	7	8	9	10	11	12
Z_k	0.0081	0.0080	0.0075	0.0078	0.0077	0.0083	0.0078	0.0080	0.0077	0.0081	0.0078	0.0082
U_k	0.0818	0.0826	0.0824	0.0789	0.0816	0.0767	0.0821	0.0753	0.0778	0.0755	0.0760	0.0832
	Workstations											
	13	14	15	16	17	18	19	20	21	22	23	24
Z_k	0.0082	0.0082	0.0080	0.0076	0.0077	0.0084	0.0077	0.0083	0.0080	0.0085	0.0076	0.0079
U_k	0 .0819	0.0782	0.0845	0.0845	0.0753	0.0794	0.0788	0.0827	0.0769	0.0799	0.0795	0.0815

10.6 Discussions

The four above experiments applied the research methodologies of the MMAS algorithm to the AOIS problem. In this section, all of the experiments are compared and discussed. The four experiments considered different experimental parameters in terms of problem size, assumptions, and optimisation methods. The aim in all of the experiments was to find the optimal inspection allocation such that the total manufacturing cost could be reduced. Table 10.22 shows the performance of the MMAS algorithm in Experiments 1 and 2 against the other methods. In these two experiments, the AOIS problem was solved with the CEM, ROT-a, b, PRS, SA, GA, PSO and MMAS methods. Table 10.23 shows the performance of the MMAS algorithm for Experiments 3 and 4 in comparison with methods in selected case studies from previous literature. It can be seen that there are more performance measures in Experiments 1 and 2 as opposed to the remaining experiments. This is because the data for the first two experiments were created for this research, thus enabling the measurement of performance measures. In particular, the optimal solution could be obtained by the CEM in Experiment 1, thus, many performance measures, such as DFOS and number of optimal solutions, were used.

The average solution quality from the applied methods is shown, as well as the standard deviation of the solution quality and the worst DFOS result. The aim of most of the experiments was to determine where the inspection operations should be located in order to minimise the total cost. The total cost is the sum of the costs of production, inspection, and failures (during production and after shipment). All of the experiments, except Experiment 2, used CEM as benchmark to test the methods. It can be seen that each experiment generated 30–100 cases to represent the features of different manufacturing systems, to a total of 580. It was assumed in all experiments that there were a limited number of inspection stations available to be allocated among the different processing workstations, ranging from 3 to 8.

Table 10.22: Performance of MMAS for Experiments1and 2

Experiment	Number of cases generated	Number of inspection stations	Number of WS	Used method	% number of optimal solution	Average total cost	Average σ Total cost	Average (DFOS)	Average σ DFOS	Worst result (DFOS)	Average processing time (s)	Savings time %
Experiment 1	100	5	12	ROT-a	0.0	605,222	45,631	0.0884	0.054	0.277	2	97
				PRS	0.0	628,819	61,589	0.1244	0.0560	0.246	3	96
				SA	65	550,347	32,678	0.0006	0.0012	0.008	45	50
				GA	25	552,335	32,874	0.0024	0.0021	0.009	55	38
				PSO	70	550,350	31,699	0.0004	0.00069	0.0028	40	55
				MMAS	79	550,186	31,586	0.0001	0.00050	0.0029	35	61
Experiment 2	100	8	24	ROT-a	n/a	2556.65	35.08	n/a	n/a	n/a	3	n/a
				ROT-b	n/a	2526.51	32.26	n/a	n/a	n/a	4	n/a
				PRS	n/a	2571.68	8.25	n/a	n/a	n/a	4	n/a
				SA	n/a	2505.561	4.09	n/a	n/a	n/a	90	n/a
				GA	n/a	2521.44	29.95	n/a	n/a	n/a	110	n/a
				PSO	n/a	2503.63	4.23	n/a	n/a	n/a	89	n/a
				MMAS	n/a	2492.76	1.56	n/a	n/a	n/a	86	n/a

n/a: not applicable because not possible to perform CEM, WS: Workstation

Table 10.23: Performance of MMAS for Experiments 3 and 4

Experiment	Number of cases generated	Number of inspection stations	Number of WS	Solution technique	Average (DFOS)	σ DFOS
Experiment 3	50	3	6	HM	0.000	0.000
				MMAS	0.000	0.000
			8	HM	0.002	0.001
				MMAS	0.000	0.000
		4	10	HM	0.003	0.005
				MMAS	0.000	0.000
			12	HM	0.0001	0.001
				MMAS	0.000	0.000
		5	14	HM	0.003	0.002
				MMAS	0.000	0.000
			16	HM	0.006	0.007
				MMAS	0.0002	0.0007
Experiment 4	50	5	13	GA	0.0041	n/a
				MMAS	0.000	n/a

In addition, each experiment tested a different number of workstations, ranging from 6 to 24. This range of workstations generated a range of feasible solutions from 64 to 16,777,216. It is well known that as the number of workstations increases, the number of feasible solutions increases significantly. Consequently, as the problem increased in size, the comparison between the MMAS algorithm and the other developed methods became based on the absolute value instead of the CEM benchmark. For these large cases, the optimal solution could not be obtained in a reasonable amount of time. In the AOIS problem, it is possible that not all inspection station plans are equivalent in terms of total cost, since some combinations of inspection stations may be economically preferable to others, most likely due to the difference in cost structures of the inspection plans and the process characteristics. The best

method is one that is able to produce the best combination of inspection stations, leading to the optimal solution, or close to the optimal solution. In other words, the main concern of these experiments is how close the solutions are to the optimal solution.

In Tables 10.22 and 10.23, some performance measures appear as (n/a), since not all performance measures were considered in all of the experiments. Most of the case studies in the experiments were interested in solution quality, particularly in terms of DFOS. It can be seen from Tables 10.22 and 10.23 that the average DFOS solution quality obtained by the MMAS algorithm is considerably better than the solutions obtained by the other methods in all the experiments except for Experiment 2. In all the experiments, the standard deviation of the total cost of inspection plans obtained by the MMAS algorithm is also significantly smaller than the other methods, indicating that the MMAS algorithm is the most reliable and effective. It was found that the MMAS algorithm reaches the optimal solution in the vast majority of the experiments. The MMAS algorithm reached the optimal solution 79% of the time in Experiment 1 and in all other experiments when the number of workstations was less than 16. However, when the number of workstations equalled 16 (i.e., in Experiment 3), the MMAS produced a solution not at, but very close to the optimal solution 0.00024.This optimal solution was obtained with CEM when the full enumeration of the search space could be generated in a reasonable time. It was also found that the worst DFOS result obtained by the MMAS algorithm was much better than the other methods. The reason for the good performance of the MMAS algorithm is due to its heuristic information, local search method and the advantage of memory. Regarding the PSO algorithm, the algorithm reached the optimal solution 70% of the time in Experiment 1. This indicates the effectiveness of the algorithm as the fitness distance correlation (FDC) obtained by the PSO was 0.57, which shows there is a good correlation between the solution quality and the distance to the optimum.

However, the SA and GA methods, in fact, use the same local search methods as the MMAS algorithm; nevertheless, these methods did not perform as well as MMAS. It was found that the average DFOS obtained by the heuristic methods in Experiments 3 was not as good as that in the MMAS algorithm. On the other hand, the average DFOS obtained from Experiment 1using GA method was much better than that in Experiment 4. The results show that the average DFOS obtained by a pure random search (PRS) algorithm was rather far from the optimal solution as compared with all the other methods. The PRS algorithm was based on placing inspection stations purely randomly and did not have any operator to guide the algorithm to good solutions. This confirms that the solutions found by the MMAS algorithm were not just lucky strikes. Also the results show that the average DFOS obtained by the ROT-a method was rather far from the optimal solution as compared with all the other methods. Yet, the ROT-a method was unable to approach the optimal solution, which was found by the MMAS, SA, GA and PSO methods in Experiment 1. The results obtained by the MMAS algorithm for 100 case studies are investigated and leading to develop new rule of thumb (ROT-b) to tackle the AOIS problem. The performance of the ROT-b is tested against the MMAS and ROT-a. It was found that the performance of ROT-b is much better than that of ROT-a, but is worse than the MMAS algorithm.

In terms of saving money on long term, it was found that the additional cost of using the MMAS algorithm to obtain near optimal solution is much less than the waiting time cost resulting of using the CEM. Also, it was found that the MMAS algorithm saving $6205 annually over the very well known method GA for engine valves case study consisting of 36 workstations. In addition, the MMAS algorithm saving $28,805 annually over the worst solution obtained by the PRS algorithm for the engine valves case study consisting of 24 workstations.

The MMAS algorithm requires less processing time to reach near optimal solutions as compared with the other studied methods. The reduction of the processing time becomes very obvious when the problem increases in size. For example, the time saved over the CEM optimisation method was found to be up to 61% when the number of feasible solutions in Experiment 1 was greater than 4,000. In the MMAS algorithm, by adapting the number of ants that apply the local search, a good compromise between convergence speed and solution quality can be obtained. On the other hand, the PSO method requires less time to reach close to the optimal solution as compared to SA and GA. It can be seen that PSO is a competitive technique to the MMAS algorithm in terms of processing time. In contrast, the execution time provided by the ROT-a method is significantly less than all the other methods. This is because the ROT-a method is based on a simple algorithm and it can reach the optimal solution in one pass. In reality, however, solution quality is of more concern to companies as opposed to the processing time, provided that the execution time is reasonable. It is concluded that the MMAS algorithm is the best approach for the AOIS problem, particularly when the number of workstations increases considerably.

Chapter 11
Conclusions and future work

This research was focused on determining the optimal allocation of inspection stations (AOIS) in serial multistage manufacturing processes. This research contributes to knowledge by developing a general cost model including all the main characteristics that are described in section 2.6 for solving the AOIS problem. In addition, the optimality was defined in terms of minimising the cost per conforming output unit accepted by the customer. The general cost model was developed under the assumption that only a limited number of inspection stations (e.g. owing to a limited budget) were available. In this chapter, the main results presented in the thesis are outlined and some directions and perspectives for future research are discussed.

The ant colony optimisation (ACO) algorithm was proposed to tackle the AOIS problem. It should be noted that the ACO approach was rather unexplored for the AOIS problem, at the time this research started. The ACO was proposed after different optimisation methods were investigated. It was found that the ACO technique has the most characteristics among the other studied methods. Different ant colony versions were studied, leading to the MMAS algorithm being proposed as a novel approach to tackle the AOIS problem. The fitness distance correlation (FDC) for the AOIS problem using the MMAS algorithm indicates strong correlation between the solution quality and the distance to the optimum. This indicates that the MMAS algorithm is well suited to the AOIS problem (see section 5.5.2). It should be noted that none of the studies surveyed in chapter 2 investigated the fitness landscape for the AOIS problem. This research provides further information about understanding the fitness landscape of the AOIS problem. MMAS is an ant colony optimisation algorithm that was designed originally to begin with a very explorative search

phase and, subsequently, to make a slow transition to an intensive exploitation of the best solutions found during the early search. Three different variants of the MMAS algorithm were developed to tackle the AOIS problem. The performance of the MMAS algorithm was compared to the complete enumeration method (CEM), genetic algorithm (GA), simulated annealing (SA), particle swarm optimisation (PSO), a pure random search algorithm (PRS) and rule of thumb (ROT-a, b), supported by the methods used by the case studies in the experiments conducted. The following subsections illustrate the types of experiments used, and the main conclusions obtained in this research.

11.1 Experimental review

Four experiments were conducted to test the performance of the MMAS algorithm. These experiments considered different experimental parameters, in terms of the size of the problem, assumptions and methods, which were used to solve the problem. The data for Experiments 1 and 2 were generated randomly, based on a uniform random distribution. However, the first experiment's experimental parameters were based on assumed values, and considered a relatively small-scale problem with 12 workstations. Within this number of workstations, the full enumeration of the search space can be generated in a reasonable time. This allows the performance of the MMAS algorithm to be tested in comparison with the optimal solution. Therefore, the comparison between the MMAS algorithm and the other developed methods, GA, SA, PSO, PRS and ROT-a, was based on deviation from the optimal solution (DFOS). In the second experiment, the experimental parameters were based on real data, and it was a larger-scale problem than Experiment 1, with 24 workstations. Hence, the comparison between the MMAS algorithm and the other developed methods was based on absolute value. The aim of the second experiment was to test the performance of the MMAS algorithm when the AOIS problem grows significantly. Both Experiments 1 and 2 used a

random sample of 100 cases to represent the characteristics of different manufacturing systems.

Experiments 3 and 4 were also used to test the performance of the developed MMAS algorithm. These experiments were based on data of case studies, which were selected from the earlier literature review. In addition, Experiments 3 and 4 are different from the previous experiments in terms of their experimental parameters, assumptions, methods and inspection strategy (100% inspection of items). These experiments considered the number of workstations, ranging from 6 to16. Therefore, the comparison between the MMAS and the methods developed by the case studies in the experiments conducted was based on DFOS. The two experiments generated the number of cases randomly, and each consisted of 50 cases. The total number of cases generated from the four experiments was 550, and these represented different characteristics of manufacturing systems. The number of workstations which were considered by the four experiments led to the generation of a number of feasible solutions in the search space, ranging from 64 to 16,777,216.

11.1.1 Local search

None of the metaheuristic methods used in the literature review used local search to improve the performance of their models. This work gives the details of an implementation of a local search method for the AOIS problem. It is widely agreed that in many of the most efficient implementations of ACO algorithms, the ants may apply a local search to improve the solutions they have constructed. In addition, the local search is part of the *DaemonActions* of the ACO algorithm. Therefore, many local search methods in other problems such as vehicle routing problems, quadratic assignment problems, travelling salesman problems and job scheduling problems have been investigated. In this research, six local search methods which are well-known in these problems are used and tested to improve the performance of the MMAS algorithm. These methods are crossover, interchange, swap, single insertion, delete

and add and block insertion. To yield a further reduction in run-time and to focus the local search around the part where potential improvements can be found, the don't look bits method was used. The experimental results show that the local search methods developed improved the performance of the MMAS algorithm considerably. This improvement can be observed in the average deviation from the optimal solution DFOS. In particular, the local search methods of crossover, single insertion and block insertion performed better than the other methods in terms of solution quality (see section 9.4).The superior results indicate the successful incorporation of the local search with the MMAS algorithm to increase the possibility of finding a better solution. This good performance can also be observed by varying the number of workstations. MMAS reached the optimal solution in certain conditions in 79% of the experiments conducted, particularly in Experiment 1(see section 10.1.6). In addition, despite the GA and SA methods that were developed and combined with the same local search method used in the MMAS algorithm, the local search method in the MMAS performed better than the other two methods. This is because the initial solutions created by the MMAS are good, and therefore the subsequent local search requires fewer steps to arrive at a good quality solution.

11.1.2 Heuristic information

It is well known that artificial ants in MMAS need heuristic information to guide them, so that they can build reasonably good solutions from the initial search of the algorithm. There is no heuristic information created for the AOIS problem. In addition, heuristic information is part of the ACO algorithm. Thus, there is a need to create heuristic information for the AOIS problem. In many cases heuristic information is the cost, or an estimate of the cost, of adding the component or connection to the solution under construction. In the AOIS problem there are many costs incurred by inspection operations or processing operations resulting from passing the raw materials through a sequence of processing workstations. These include the

costs of inspection, replacement, reworking, manufacturing, penalties and scrapping. It was found that the operation cost, defective rate and inspection cost are the most appropriate factors to consider as a guide to the heuristic information toward the best inspection plans of the search space.

This research contributes to knowledge by developing two novel heuristic methods to guide the ant to locate an inspection station to a workstation based on the concerns of operation cost (U_k), inspection cost (IC_k) and defective rate (Z_k), respectively. These are the Operation Cost and Defective rate Method (OCDM) and the Scores Method (SM). The MMAS algorithm with two types of heuristic information $MMAS_{\text{-OCDM}}$ and $MMAS_{\text{-SM}}$ and with no heuristic information $MMAS_{\text{-NH}}$ was applied to 50 different cases generated randomly from the general cost model case study. The experimental results determined that the performance of the MMAS algorithm, in terms of solution quality when using heuristic information, was significantly better than without using heuristic information (see section 9.3). Specifically, it was found that the best average DFOS obtained by the $MMAS_{\text{-OCDM}}$ and $MMAS_{\text{-SM}}$ were 0.00015 and 0.00033 respectively. However, when no heuristic information $MMAS_{\text{-NH}}$ was used, the best average DFOS was 0.0186. It is concluded that it is important to use heuristic information with the MMAS algorithm for tackling the AOIS problem.

11.1.3 MMAS tuning parameters

Selecting relevant parameters in the MMAS can have a great impact on the algorithm's performance, and a good combination of parameters will increase the overall search capability and convergence of the algorithm. On the other hand, inappropriate values will certainly slow down the process of discovering the best solution to the AOIS problem, and may even prevent the algorithm from finding the best solution. The optimal combination of the most influential parameters for the MMAS algorithm, applied to 50 case studies, was

identified. These cases were generated randomly using a uniform distribution, in order to represent the characteristics of different manufacturing systems. The MMAS algorithm was applied to these cases for each triplet (α, β, ρ) of parameter settings, in order to optimise the average DFOS. It was found that these optimal parameters significantly improved the performance of the MMAS algorithm (see section 8.4.1). In addition, the experimental results determined that the performance of the MMAS, GA, SA and PSO algorithms depends on the appropriate setting of parameters. For a specific problem, the optimal parameters will be slightly different, but the suggested parameters should be good for many AOIS problems with a similar structure.

11.1.4 MMAS behaviour

Measurement of convergence of the MMAS algorithm is important to stop the algorithm needlessly running when the optimum solution has already been achieved. To investigate the convergence of the MMAS algorithm towards a near optimal solution, extensive experiments were conducted. These experiments tested the most influential parameters α, β and ρ of the MMAS algorithm. In addition, as the MMAS algorithm relies on a number of parameters that control its behaviour, sensitivity analysis for the most influential parameters α, β and ρ was used. The aim was to determine how "sensitive" the model is to changes in the value of the parameters of the model. Parameter sensitivity is usually performed as a series of tests in which the modeller sets different parameter values to see how a change in the parameter causes a change in the behaviour of the MMAS algorithm.

The results determined that the ratio between α and β was the key driver of the convergence speed of the results. Since α and β are the parameters of relative influence of the pheromone strength and the heuristic information respectively, the relative influence of the pheromone strength which dominated in the searching process makes the convergence happen early. The

results showed the optimum ratio between α and β ($\alpha/\beta = 5/2$). It was found that when $\alpha > \beta$ (e.g. $\alpha/\beta = 5/1$) the convergence speed is faster. Correspondingly, there was no convergence when $\beta > \alpha$ (e.g. $\alpha/\beta = 1/3$). Regarding the ρ parameter (evaporation rate),it was found that if ρ is large (e.g. $\rho = 0.05$), it is easy to find marked relative differences between the pheromone trails on arcs contained in high quality tours and those which are not part of the best tours in a few iterations, so the algorithm may stagnate and prematurely converge. Otherwise, for a lower ρ (e.g. $\rho = 0.01$), the pheromone trails on arcs which do not belong to the high quality tours will not decrease faster and the algorithm is able to explore a wider search space, but longer evolution iterations are needed. Therefore, if a larger total evolution iteration is used, a lower ρ can be selected for obtaining a better convergence value; otherwise, a higher ρ will help achieve a better convergence speed (see section 9.1).

Regarding sensitivity analysis, it should be noted that the vast majority of papers reviewed did not consider this issue. This research provides further information about understanding the sensitivity analysis of the AOIS problem. As MMAS algorithm rely on a number of user-defined parameters that control their behaviour, sensitivity analysis for the most influential parameters α, β and ρ was conducted. The aim was to study the importance of each parameter for increasing the number of workstations in the AOIS problem. Four different sizes of AOIS problem are tested, with 15, 16, 17 and 18 workstations. The experimental results show that the studied parameters α, β and ρ do not have a significant influence on the performance of the MMAS algorithm when the number of workstations is increased (see section 9.2). The experimental results show that from a sensitivity perspective the parameters α and β are sensitive parameters to a fixed cost model.

Updating the pheromone trails for the MMAS algorithm was done using a mixed strategy s^{ib+gb}. The aim of this strategy was to obtain stronger exploration of the search space early

in the search and stronger exploitation of the overall best solution later in the run. This mixed strategy specifies that in the first 300 iterations, the iteration best ant s^{ib} was used to update the pheromone trails, and then, at every tenth iteration, the global best ant s^{gb} was used for the pheromone trail update. To test the performance of this mixed strategy, it was compared with the other two methods for updating pheromone trails, s^{ib} and s^{gb}. The results showed that the performance of the MMAS in terms of average deviation from the optimal solution when using mixed strategy is much better than the other two strategies (see section 9.6). Specifically, it was found that the average DFOS obtained by the MMAS when the mixed strategy was used was 0.00012, but when the iteration best ant and the global best ant were used, the average DFOS were 0.00015 and 0.0002 respectively.

In this research, three variants of the MMAS algorithm were developed and tested. In the first variant, denoted as $MMAS_{10+ls}$, 10 ants were used and every ant applied local search to its tour. In the second variant, $MMAS_{+ls+ib}$, the algorithm started with a fixed number of 10 ants and then the number of ants which applied local search was then successively increased by one after a certain number of iterations. The third variant uses the MMAS algorithm without local search and is denoted as $MMAS_{-nls}$. The results showed that the best average DFOS obtained by the $MMAS_{10+ls}$ algorithm was 0.00012, whereas the best average DFOS obtained by the $MMAS_{+ls+ib}$ and $MMAS_{-nls}$algorithms were 0.0002 and 0.005 respectively (see section 9.7). It was concluded that the $MMAS_{10+ls}$ algorithm performs better than the other two variants in terms of solution quality.

On the other hand, the $MMAS_{+ls+ib}$ algorithm needed less processing time to reach a near-optimal solution compared with the other two variants. It was found that by adapting the number of ants which apply local search, a good compromise between convergence speed and solution quality can be obtained. The computational results presented in this section

suggest that, especially for larger problems, using the $MMAS_{+ls+ib}$ algorithm may be advantageous. In the variant $MMAS_{-nls}$ algorithm, the processing time was slightly higher than with the other two variants.

11.1.5 Major results

The results showed that the MMAS algorithm reached the optimal solution in the vast majority of the experiments conducted. In Experiments 1 and 2, the MMAS algorithm performed much better than the other methods. The optimal solution of the experiments was obtained by using a complete enumeration method. This is because the full enumeration of the search space can be generated in a reasonable time. The experimental results confirmed that the average DFOS obtained by the MMAS algorithm was significantly better than the solutions obtained by the other methods, in all the experiments. In addition, in all the experiments, the standard deviation of the total cost of inspection plans obtained by the MMAS was much less than the other methods. This indicates a more reliable performance of the MMAS algorithm. In Experiment 1, it was found that even the worst result of the DFOS obtained by the MMAS algorithm is much better than the other methods developed.

The results showed that the PSO method developed reached an optimal solution in 70% of the experiments conducted. This percentage was lower than the MMAS algorithm, which achieved 79% for the same number of experiments. In addition, the solution quality, in terms of average DFOS, achieved by the PSO method was close to the optimal solution. PSO therefore comes second among the developed methods in terms of solution quality, and as a result, the PSO method could be considered a competitive technique to the MMAS algorithm. On the other hand, the results of the experiments showed that SA and GA reached the optimal solution in 65% and 25% of experiments respectively. In particular, the performance of GA for tackling the AOIS problem, in terms of solution quality, was much poorer than the

MMAS and PSO. Apart from the PRS algorithm, the solution quality obtained by the ROT-a method was lower than the other methods studied, particularly in comparison with the MMAS algorithm. The solution quality obtained by the PRS algorithm was thus the worst of all the methods studied, particularly in comparison with the MMAS algorithm, because the solutions obtained were purely random. The results obtained by the MMAS algorithm for 100 case studies were extensively investigated, leading to the development of a new rule of thumb (ROT-b) to tackle the AOIS problem. The performance of ROT-b was tested against ROT-a and the MMAS algorithm. The three algorithms were applied to the same 100 case studies for engine valves consisting of 24 workstations. The average total costs obtained by the ROT-a, ROT-b and the MMAS algorithm were 2556.65, 2526.51 and 2492.76 respectively. It was concluded that the performance of ROT-b in terms of solution quality was much better than that of ROT-a, but was worse than the MMAS algorithm. The rule of thumb developed will be very useful for industry, as no tuning parameters are needed.

Regarding saving money in the long term, it was found that the additional cost of using the MMAS algorithm to obtain a near optimal solution is much less than the waiting time cost resulting from using CEM for the engine valves case study consisting of 24 workstations. Also, it was found that the MMAS algorithm costs $6205 less annually than the GA for the engine valves case study consisting of 36 workstations. In addition, the MMAS algorithm costs $28,805 less annually than the PRS algorithm for the engine valves case study consisting of 24 workstations. It was concluded that using the MMAS saves money by minimising the computation time and the total cost of the product, and keeps the company in a good competitive position.

In terms of processing time, all the developed methods applied in Experiments 1 and 2 were tested by using CPU time. This is the time taken to execute the computer programs for problem-solving in the computer system. It should be noted that an efficient method should

provide a significant saving in execution time over the complete enumeration method. The experimental results confirmed that, except for the ROT method, the MMAS algorithm needed less processing time to reach a near optimal solution than the other optimisation methods. This is very clear, as the AOIS problem increases significantly with the number of processing workstations. In contrast, the execution time provided by the ROT-a method was significantly less than the other methods used. This is because the ROT-a method is based on a simple algorithm, and the solution can be obtained in just one iteration. However, the solution quality obtained by the ROT-a method was worse than the other methods. In real life, the production management in firms is more interested in solution quality than processing time, provided the execution time of the method used is reasonable. On the other hand, the processing time needed by the GA to reach a solution close to the optimal was much greater than the other methods. It is well known that the main drawback of the GA is that it requires a large number of response (fitness) function evaluations depending on the number of individuals and the number of generations. Therefore, genetic algorithms may take a long time to evaluate the individuals.

In summary, the strength of the MMAS algorithm was demonstrated in its considerably shorter execution time and robustness. As a result, the MMAS algorithm is found to be a proper approach for tackling the AOIS problem, even when the number of workstations increases significantly.

11.2 Future work

For future research the following possible area may be studied:

- The incorporation of the material handling cost in the case of inspection not immediately performed after the processing workstation but at a special inspection location. The material handling cost is incorporated explicitly into the problem to make it more realistic.

The material handling cost is a function of distance travelled, type of handling equipment, type of product, inspection station layout. Therefore, in this case, the cost of inspection will include the cost of physical inspection and the cost of material handling.

- Online tuning of the MMAS algorithm is an alternative to offline tuning. Based on the results obtained, the parameters given here for the AOIS problem applications performed very well over a wide range of instances. Nevertheless, in other applications, adaptive versions, which tune the parameters dynamically during the algorithm's execution, may increase the algorithm's robustness. Typically, this consists of the modification of an algorithm's parameter settings while solving a problem instance. A potential advantage of an online modification of parameters is that algorithms may adapt better to the particular instance's characteristics.
- The ant system and its variants have been applied to a variety of discrete and continuous optimisation. However, some famous optimisation problems have not been tackled with ant algorithms yet such as material requirements planning (MRP) and multi-objective problems. This is most often as a result of difficulties in finding a representation of the solution space that can be travelled by ants. Based on the results obtained in this research, further research is needed to find solutions to these difficulties, and apply the ACO to an ever increasing range of problems.

11.3 Contribution of the thesis

This thesis delivers a number of contributions to the field of allocation of inspection stations and development of optimisation algorithms, especially to the area of Ant Algorithms.

11.3.1 General cost modelling formulation

The cost of the AOIS problem involves many characteristics, including inspection errors (type I and type II), internal failure cost (rework and scrap), external failure cost (repair and

replacement), inspection cost (fixed and variable) and manufacturing cost. No literature has previously considered all these characteristics together. This is because many earlier studies were more interested in developing new heuristic methods to approach the complexity of the AOIS problem. These simplified assumptions are also introduced to allow a tractable formulation model and solution. The simplified assumptions in those cost models in the literature review have led to the lack of generality. All the cost models surveyed assumed that when an inspection is performed after the processing workstation, 100% inspection occurs. This assumption increases the cost of inspection and inspection time. In addition, all the cost models studied used the total cost per input unit and the total cost per output unit as the objective function. However, a customer totally sophisticated in quality determination is hypothesised, that is, one who can determine the quality of an item with 100% accuracy. Furthermore, all previous cost models were represented external failure cost items as aggregate or only include one of them.

This research contributes to knowledge by developing a general cost model (GCM) to include all the characteristics of the AOIS problem described above. The developed cost model also contributes to knowledge by determining the locations of inspection stations using the sampling inspection plan. Furthermore, the GCM is developed such that the optimality is defined in terms of minimising the cost per item accepted by the customer. To do so, the general cost model is developed such that the number of conforming parts can be computed at each processing workstation. Also the external failure costs are represented to be more complex to include all of its items. Introducing all these issues into the GCM significantly increases the level of complexity, makes the general cost model is very different from those in the previous literature and maintains the generality. Figure 11.1 shows the contribution of the GCM in serial multistage manufacturing processes.

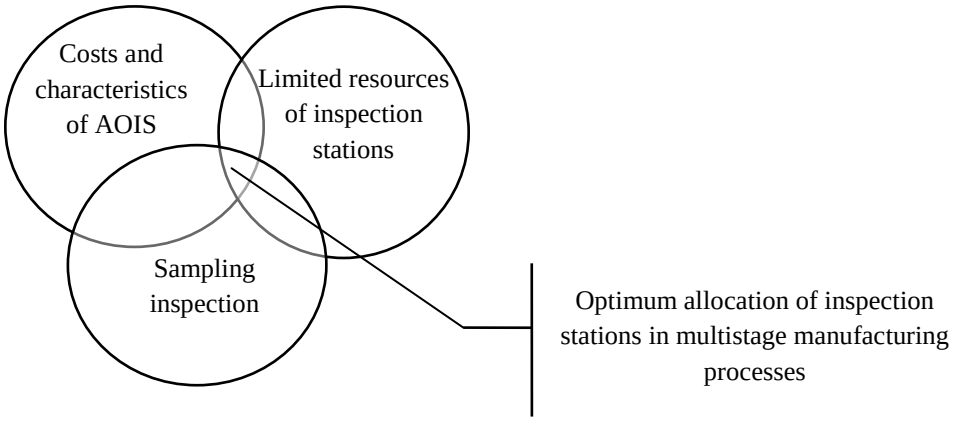


Figure 11.1: Contribution of the GCM in serial multistage manufacturing processes

11.3.2 Evaluating the MMAS algorithm on the AOIS problem

A max-min ant system (MMAS) algorithm was evaluated against a new type of problems known as the allocation of inspection stations (AOIS) in serial multistage manufacturing processes. This was a problem because, in the allocation problem, when the number of processing workstations increases, the processing time required for solving the problem grows exponentially, and the complete enumeration method becomes impractical. Empirical validation shows that the MMAS algorithm is effective and efficient compared to other metaheuristics algorithms. A method has been developed to find the optimal combination of the most influential parameter values for the MMAS algorithm. This work is important because it shows the link between the theoretical and practical algorithms being developed in the field. Also, this work provides further information on optimisation by evaluating MMAS algorithm on this type of AOIS problem in a serial multistage manufacturing process.

11.3.3 Developing two heuristic information methods

Two heuristics information methods for the problem of allocating inspection stations have been constructed. The two novel heuristic methods are created to guide the ant to locate an inspection station at a workstation. These are the Operation Cost and Defective rate Method

(OCDM) and the Scores Method (SM). The originality of the contribution is that these methods have never been applied to this type of AOIS problem or in different areas. In particular, the SM was first used by Shetwan et al. (2011) as a new heuristic method to solve the AOIS problem as a part of this research. Previous heuristic information methods were based on a simple idea such as in TSP. However, the concept of the two heuristic methods is more complex and very different from those in the previous literature. Specifically, the steps of SM include investigation of the characteristics for the AOIS problem, selecting the most important characteristics, assigning scores for each workstation, finding the total scores, compute the priority, link the priority via mathematical formula and implementing the heuristic information method. The significance of the contribution is that the heuristic information makes ACO algorithms (AS, ACS, Ant-Q and MMAS) more efficient in solving real-world problems in a number of different areas of the AOIS problem. Examples are the rigor of the inspections (acceptance limits) for each inspection station, the number of inspections executed (sample size-sampling frequency) for each inspection station and these issues are able to include different production configuration such as assembly and non-serial. Furthermore, by introducing heuristic information the probable search space becomes much smaller than the original search space. In addition, the heuristic information increases the ability of ACO to find high-quality solutions in a reasonable time.

11.3.4 Developing a good new rule of thumb

A good new rule of thumb has been developed to solve the AOIS problem. The solutions obtained by the MMAS algorithm for solving the AOIS problem are extensively investigated. Investigation of inspection plans is done by looking at the positions of inspection stations in the solutions obtained and comparing them with their characteristics. The aim is to analyse the behaviour of the MMAS algorithm for allocating inspection stations. The analysis of the solutions leads to the development of a good new rule of thumb. The importance of the rule

of thumb lies in the simplicity of its method: no tuning parameters are needed, so it is preferred by industry.

Appendix A

This appendix describes the exact methods used in the literature review.

1. Integer programming

An integer programming problem is any linear programming model in which some or all the variables are restricted to be integral (Hillier and Lieberman, 2010). In its most general form, the aim in such a program is to assign integers to a set of variables such that a set of linear inequalities are satisfied and a linear goal function is minimised or maximised. It is widely known that many real-world problems can be captured conveniently by integer programs (Jonsson and Nordh, 2006). An integer programming problem can be formulated as follows:

$$\begin{aligned} &\underset{x}{Max} \quad c^T x \\ &\text{subject to} \\ &Ax \le b \\ &x \ge 0, \quad x \text{ integer} \end{aligned}$$

where x is the vector of variables to be solved for, A is a matrix of known coefficients, and c and b are vectors of known coefficients. The expression 'cx' is called the objective function, and the equations '$Ax \le b$' are called the constraints.

2. Linear programming

Linear programming is a technique for optimisation models in which the objective and constraints functions are strictly linear. Linear programming is one of the most successful disciplines within the field of operation research (Hamdy, 2003). In its standard form, the linear programming problem calls for finding non-negative $x_1,....,x_n$ so as to maximise a linear function $\sum_{j=1}^{n} c_j x_j$ subject to a system of linear equations:

$$a_{11} x_1 + ...+ a_{1n} x_n = b_1$$

$$\vdots$$

$$a_{m1} x_1 + ...+ a_{m\,n} x_n = b_m$$

This problem can be stated in vector notation as:

$$\begin{aligned} &\text{Maximise } c^T x \\ &\text{subject to} \quad Ax = b \\ &\qquad\qquad\quad x \ge 0 \end{aligned}$$

where $A \in R^{m \times n}$ is assumed to have linearly independent rows, and $b \in R^m$ and c, $x \in R^n$. In fact, any problem of maximising or minimising a linear function subject to linear equation and equalities can be easily reduced to the standard form (Megiddo, 1991).

3. Non-linear programming

Nonlinear programming is the process of solving a system of equalities and inequalities over a set of unknown real variables, along with an objective function to be maximised or minimised (Hamdy, 2003). In general, the *non-linear* programming problem is to find $x = (x_1, x_2, ..., x_n)$ so as to

maximise $f(x)$,

subject to

$g_i(x) \leq b_i$, for $i = 1,2,...,n$,

and $x \geq 0$,

where $f(x)$ and the $g_i(x)$ are given functions of the n decision variables. There are many different types of nonlinear programming problems, depending on the characteristics of the $f(x)$ and $g_i(x)$ functions (Hillier and Lieberman, 2010).

4. Branch and Bound

Branch and Bound algorithm is a general algorithm for finding optimal solutions of various optimisation and combinatorial optimisation problems. In particular, the idea of branch-and-bound algorithms is based on partitioning the solution set and using lower bounds to construct a proof of optimality without an exhaustive search. That is, a search tree is built, where each node represents a partition of the set of solutions, and each child of a node is a subset of that partition (this is the 'branch' part). An algorithm is available for calculating a lower bound on the cost of any solution in a given subset (the 'bound' part). The search tree is searched by using the lower bounds to remove whole sub-trees, effectively reducing the number of checked solutions. Clearly, the partitioning and the way the lower bound is defined have a major impact on the performance of the algorithm (Di Caro, 2004).

5. Dynamic programming

Dynamic programming is a recursive method, which determines the optimum solution to an n-variable problem by decomposing it into n stages, with each stage constituting a single variable sub-problem. Each stage has a number of states associated with the beginning of that stage. The number of states may be either finite or infinite. Thus the full recursion tree

generally has polynomial depth and an exponential number of nodes (Hamdy, 2003). Dynamic programming works at best by computing the value of each state in an effective way, and using these values to compute the optimal decision policy.

Assume that decision variables x_n (n=1, 2, 3, 4) be the immediate destination on stage n for a problem consisting of 4 stages. Given that s is state and x_n as the immediate destination. Given s and n, let x_{n*} denote any value of x_n (not necessarily unique) that minimizes $f_n(s, x_n)$, and let $f\,n^*(s)$ be the corresponding minimum value of $fn(s, x_n)$. Let $f_n(s, x_n)$ be the total cost of the best overall policy for the remaining stages as shown in equation C1:

$$f_{n^*(s)} = \min_{x_n}\{c_{sx_n} + f_{n^*+1}(x_n)\} \qquad \text{C1}$$

The value of cs_{xn} is the cost to move from state i to state j, which is denoted by c_{ij}, by setting $i = s$ (the current state) and $j = x_n$ (the immediate destination).

Therefore, finding the optimal policy decision starting in state s, at stage n, requires finding the minimum value of x_n. The recursive relationship among dynamic programming problems is summarised below:

N = number of stages.

n = represents current stage (n=1, 2,…, N).

s_n = current state for stage n.

x_n = decision variable for stage n.

x_n^* = optimal value of x_n (given s_n).

c_{sxn}= constant.

$f_n(s_n, x_n)$ = contribution of stages n, n+1,…,N to objective function if system starts in state s_n at stage n, immediate decision is x_n , and optimal decisions are made after, as given by equation C2:

$$f_{n^*(s_n)} = f_n(x_n, x_n^*) \qquad \text{C2}$$

The recursive relationship will always be in the form given by equation C3:

$$f_n^*(s_n)_{x_n} = \min\{f_n(s_n, x_n)\} \qquad \text{C3}$$

where $f_n(s_n, x_n)$ would be written in terms of s_n, x_n , $f_{n+1}^*(s_{n+1})$, and some measure of the immediate contribution of x_n to the objective function. With the addition of $f_{n+1}^*(s_{n+1})$ on the

right–hand side, $f_{n*}(s_n)$ is defined in terms of $f^{*}_{n+1}(s_{n+1})$, which makes the expression for $f_{n^*}(s_n)$ a recursive relationship. The solution procedure starts at the end and moves backwards, stage by stage, recursively, time finding the optimal policy for that stage until it finds the optimal policy, starting at the initial stage. This optimal policy yields an optimal solution for the entire problem (Hillier and Lieberman, 2010).

Appendix B

This appendix contains the steps which describe simulated annealing procedure and pseudo-code. The basic steps of a SA algorithm are as follows:

Step1 Generate an initial feasible solution for the problem (inspection plans). This is the first current solution. Set the initial temperature, number of repetitions at each temperature step, the rule for decreasing the value of temperatures, the number of transitions at each temperature, and the time at which annealing should be stopped are referred to as the cooling schedule.

Step2 Generate a new set of solution/s from the current solution.

Step3 Evaluate the solution in terms of the objective function. Keep track of the best solution found so far.

Step4 If the newly generated solution is better than the current solution update the current solution to the newly found better solution. If not, the new solution is accepted depending on the *p*[*accep*t] Metropolis's criterion (Metropolis et al., 1953). In practice, this probabilistic acceptance is achieved by generating a uniformly random number *R* in [0, 1] and comparing it with *p*[*accept*]. If *R* < *P[accept]*, the newly solution is accepted and becomes the current solution, otherwise the newly solution is rejected and the current solution stays the same.

Step5 Iterate through steps 2, 3and 4 for the number of transitions at each temperature.

Step6 Adjust the cooling temperature using the decided reduction criterion. Iterate through steps 2, 3, 4 and 5 for the decided number of times. After ending the last iteration, stop. Print the best solution found so far.

Appendix C

This appendix contains the steps which describe genetic algorithm procedure and pseudo-code. The following steps describe the genetic algorithm procedure:

Step1 Generate a set of initial chromosomes (solutions randomly) with a population size, which is the quantity of chromosomes used in the study (inspection plans). In this research, initial populations of 50 inspection plans randomly generated are deemed to be appropriate (Langner et al., 2002). The fitness function is evaluated to each individual solution.

Step2 Roulette wheel selection method is used to reproduce the higher performance chromosomes and place them into a mating pool.

Step3 Recombination is performed based on a crossover rate which is the percentage of chromosomes in the mating pool to perform recombination or exchange. Two pieces of chromosomes in the mating pool is selected in a random way. Using the One-Point-Crossover method, randomly determine the crossover point. All genes beyond this point in the chromosome string are swapped between the two parent chromosome strings.

Step4 According to a mutation rate, which is the percentage of chromosomes in the mating pool to perform mutation, this number of chromosome in the mating pool is selected randomly. Find a mutation point in these chromosome strings at random, and then change the gene at this position. This step is performed to avoid local optima.

Step5 A new population is developed through reproduction, crossover, and mutation, and then the fitness function value of all the chromosomes can be calculated. Compare the chromosome having the best fitness function value with that in the last generation, and select the chromosome having the best fitness function value to keep for the next generation.

Step6 If the execution has not yet reached the end condition, then repeat Steps 2 to 5. A generation number is set for stopping the procedure.

Appendix D

The results for different parameter combinations for optimising GA are presented in Tables D1-D5.

Table D1: Average *DFOS for GA (Cross over 0.1)

Population	Mutation rate							
	0.001	0.002	0.003	0.004	0.005	0.01	0.017	0.02
20	14.1	13.9	14.6	14.0	14.0	14.10	14.2	14.2
30	13.7	13.5	13.3	13.5	13.5	12.9	13.6	14.0
40	13.4	13.8	13.5	13.9	13.4	14.0	14.2	13.4
50	13.2	13.3	12.6	13.0	13.0	13.5	13.4	13.5
60	12.7	13.3	13.1	13.3	13.1	13.4	13.2	12.9
70	12.9	12.5	12.9	12.5	12.7	12.7	12.9	12.7
80	12.6	12.2	12.2	12.6	12.0	12.4	12.3	12.4
90	14.1	14.0	14.0	13.9	14.6	14.10	14.2	14.2
100	12.9	13.6	14.0	13.5	13.5	13.7	13.5	13.3

*DFOS×0.001

Table D2: Average *DFOS for GA (Cross over 0.2)

Population	Mutation rate							
	0.001	0.002	0.003	0.004	0.005	0.01	0.017	0.02
20	11.7	11.8	11.8	11.9	11.8	12.2	11.8	11.8
30	11.7	11.8	11.9	11.7	12.1	11.6	11.9	11.6
40	12	11.8	11.8	11.7	11.6	11.9	11.6	12
50	11.6	11.4	11.7	11.8	11.7	11.9	11.6	11.
60	8.8	8.6	8.4	8.7	8.2	8.2	8.8	8.5
70	8.2	8.0	7.9	7.6	7.5	8.2	7.6	8.0
80	8.1	7.8	7.7	8.3	7.6	8.1	7.9	7.8
90	8.0	7.6	8.0	7.9	7.6	8.0	7.9	8.0
100	8.3	7.6	8.1	8.1	7.6	7.7	8.3	7.6

*DFOS×0.001

Table D3: Average *DFOS for GA (Cross over 0.6)

Population	Mutation rate							
	0.001	0.002	0.003	0.004	0.005	0.01	0.017	0.02
20	7.7	7.6	7.6	7.5	70	7.4	7.5	7.6
30	7.1	7.0	7.1	7.1	6.7	6.8	7.0	7.1
40	7.0	7.0	7.1	7.1	6.7	6.7	6.8	7.2
50	7.3	7.0	7.0	7.1	6.7	7.2	7.0	7.1
60	7.2	7.1	7.2	7.0	6.9	6.7	7.0	7.1
70	6.8	6.7	6.9	6.7	6.7	6.8	6.9	6.9
80	5.8	6.1	5.9	5.5	5.3	6.0	5.7	5.8
90	6.2	5.8	6.1	6.1	5.7	6.2	5.9	6.4
100	6.6	6.2	5.8	6.0	5.4	5.8	6.0	6.1

*DFOS×0.001

Table D4: Average *DFOS for GA (Cross over 0.8)

Population	Mutation rate							
	0.001	0.002	0.003	0.004	0.005	0.01	0.017	0.02
20	5.3	4.7	4.8	4.9	4.6	4.9	4.7	4.8
30	3.7	3.8	4.0	3.8	3.9	3.6	4.2	4.2
40	2.1	3.3	2.1	3.3	2.1	3.3	2.1	3.3
50	2.1	3.3	2.1	3.3	2.1	3.3	2.1	3.3
60	2.1	2.1	3.3	2.1	2.0	2.1	3.3	2.1
70	1.9	1.9	2	1.8	1.8	1.9	2.0	2.0
80	1.5	1.5	1.4	1.3	1.2	1.3	1.4	1.4
90	3.3	2.1	3.3	3.3	2.1	3.3	2.1	3.3
100	2.1	3.3	3.3	3.3	2.1	3.3	2.1	3.3

*DFOS×0.001

Table D5: Average *DFOS for GA (Cross over 0.9)

Population	Mutation rate							
	0.001	0.002	0.003	0.004	0.005	0.01	0.017	0.02
20	2.5	2.6	2.6	2.4	2.4	2.5	2.6	2.6
30	2.4	2.5	2.6	2.2	2.4	2.3	2.6	2.6
40	2.4	2.3	2.6	2.2	2.3	2.3	2.6	2.6
50	2.4	2.3	2.6	2.6	2.2	2.3	2.6	2.6
60	2.4	2.3	2.6	2.6	2.2	2.3	2.6	2.6
70	2.6	2.5	2.4	2.4	2.1	2.5	2.4	2.5
80	2.5	2.4	2.4	2.3	2.0	2.3	2.6	2.6
90	2.6	2.9	2.7	2.5	2.5	2.6	2.7	2.9
100	2.9	2.7	2.7	2.6	2.6	2.9	2.9	2.9

*DFOS×0.001

Appendix E

The results for different parameter combinations for optimising SA are presented in Table E1.

Table E1: Average *DFOS for SA

Iterations at each temperature	Temperature decrement								
	0.4	0.5	0.6	0.65	0.7	0.75	0.8	0.85	0.9
100	1.5	1.5	1.5	1.5	1.4	1.4	1.4	1.3	1.4
200	1.3	1.3	1.2	1.3	1.2	1.1	0.95	0.94	0.95
300	1.3	1.2	1.2	1.2	1.1	1.1	0.94	0.9	0.95
400	1.3	1.3	1.2	1.2	1.2	1.1	0.95	0.93	0.98
500	1.3	1.3	1.3	1.2	1.2	1.2	0.98	0.94	0.96
600	1.4	1.4	1.3	1.2	1.2	1.1	1.0	0.95	1.0
700	1.3	1.3	1.2	1.2	1.2	1.1	1.1	0.98	1.1
800	1.3	1.3	1.3	1.3	1.2	1.2	1.2	1.0	1.2
900	1.4	1.4	1.4	1.4	1.2	1.2	1.2	1.1	1.2
1000	1.4	1.4	1.3	1.3	1.3	1.3	1.3	1.2	1.3
1100	1.4	1.4	1.4	1.4	1.4	1.3	1.3	1.3	1.4
1200	1.5	1.5	1.5	1.4	1.4	1.4	1.5	1.3	1.5

*DFOS×0.001

Appendix F

The results for different parameter combinations for optimising PSO are presented in Tables F1-F4.

Table F1: Average *DFOS for PSO (ω=0.8)

Number of Particles	Cognitive coefficients											
	c_1=1 c_2=1	c_1=1 c_2=1.5	c_1=1.5 c_2=1	c_1=1.5 c_2=1.5	c_1=2 c_2=2	c_1=2 c_2=2.5	c_1=2.5 c_2=2	c_1=2.5 c_2=2.5	c_1=3 c_2=3	c_1=3 c_2=3.5	c_1=3.5 c_2=3	c_1=3.5 c_2=3.5
10	2.5	2.5	2.5	2.1	2.1	2.3	2.4	2.4	2.4	2.4	2.5	2.5
20	2.5	2.5	2.5	2.1	2.0	2.3	2.3	2.4	2.4	2.5	2.5	2.5
40	2.1	2.1	2.1	2.1	2.0	2.2	2.2	2.2	2.4	2.4	2.4	2.4
60	1.9	1.9	1.9	1.8	1.75	1.8	1.8	1.9	1.9	1.9	1.9	2.1
80	1.9	1.9	1.9	1.8	1.8	1.8	1.8	1.9	1.9	1.9	1.9	1.9
100	2.1	2.1	1.9	1.8	1.8	1.8	1.8	1.9	1.9	2.0	2.0	2.1

*DFOS×0.001

Table F2: Average *DFOS for PSO (ω=0.9)

Number of Particles	Cognitive coefficients											
	c_1=1 c_2=1	c_1=1 c_2=1.5	c_1=1.5 c_2=1	c_1=1.5 c_2=1.5	c_1=2 c_2=2	c_1=2 c_2=2.5	c_1=2.5 c_2=2	c_1=2.5 c_2=2.5	c_1=3 c_2=3	c_1=3 c_2=3.5	c_1=3.5 c_2=3	c_1=3.5 c_2=3.5
10	1.2	1.2	1.2	1.2	1	1.3	1.3	1.3	1.3	1.3	1.3	1.4
20	1.2	1.2	1.2	0.98	0.98	1.1	1.1	1.1	1.1	1.1	1.1	1.2
40	0.97	0.97	0.97	0.93	0.85	0.82	0.82	0.83	0.84	0.84	0.84	0.85
60	0.83	0.82	0.82	0.82	0.78	0.8	0.79	0.79	0.8	0.8	0.81	0.85
80	0.82	0.83	0.83	0.82	0.8	0.82	0.83	0.83	0.84	0.84	0.86	0.86
100	0.84	0.84	0.84	0.84	0.8	0.84	0.84	0.84	0.84	0.84	0.86	0.87

*DFOS×0.001

Table F3: Average *DFOS for PSO (ω=1.1)

Number of Particles	Cognitive coefficients											
	c_1=1 c_2=1	c_1=1 c_2=1.5	c_1=1.5 c_2=1	c_1=1.5 c_2=1.5	c_1=2 c_2=2	c_1=2 c_2=2.5	c_1=2.5 c_2=2	c_1=2.5 c_2=2.5	c_1=3 c_2=3	c_1=3 c_2=3.5	c_1=3.5 c_2=3	c_1=3.5 c_2=3.5
10	1.1	1.1	1.1	1.2	1	1.1	1.2	1.2	1.3	1.3	1.3	1.4
20	1.1	1.1	0.97	0.95	0.92	0.93	0.94	0.94	0.98	1.1	1.1	1.2
40	0.83	0.82	0.82	0.83	0.8	0.82	0.82	0.83	0.8	0.82	0.83	0.85
60	0.83	0.82	0.82	0.8	0.8	0.75	0.79	0.79	0.8	0.8	0.81	0.85
80	0.82	0.83	0.83	0.82	0.7	0.82	0.83	0.83	0.84	0.84	0.86	0.86
100	0.84	0.84	0.84	0.84	0.8	0.84	0.84	0.84	0.84	0.84	0.86	0.87

*DFOS×0.001

Table F4: Average *DFOS for PSO (ω=1.2)

Number of Particles	Cognitive coefficients											
	c_1=1 c_2=1	c_1=1 c_2=1.5	c_1=1.5 c_2=1	c_1=1.5 c_2=1.5	c_1=2 c_2=2	c_1=2 c_2=2.5	c_1=2.5 c_2=2	c_1=2.5 c_2=2.5	c_1=3 c_2=3	c_1=3 c_2=3.5	c_1=3.5 c_2=3	c_1=3.5 c_2=3.5
10	1.1	1.1	1.1	1.2	1	1.1	1.2	1.2	1.3	1.3	1.3	1.4
20	1.1	1.1	1.1	1.1	0.98	1.0	1.1	1.1	1.1	1.1	1.1	1.2
40	1.1	1.2	1.2	1.2	0.97	0.92	0.92	0.95	0.95	0.97	0.97	0.97
60	0.94	0.94	0.94	0.9	0.85	0.94	0.94	0.94	0.94	0.94	0.94	0.95
80	0.96	0.96	0.96	0.9	0.88	0.94	0.94	0.94	0.95	0.95	0.95	0.95
100	094	0.94	0.94	0.95	0.90	0.94	0.94	0.94	0.94	0.94	0.96	0.96

*DFOS×0.001

Appendix G

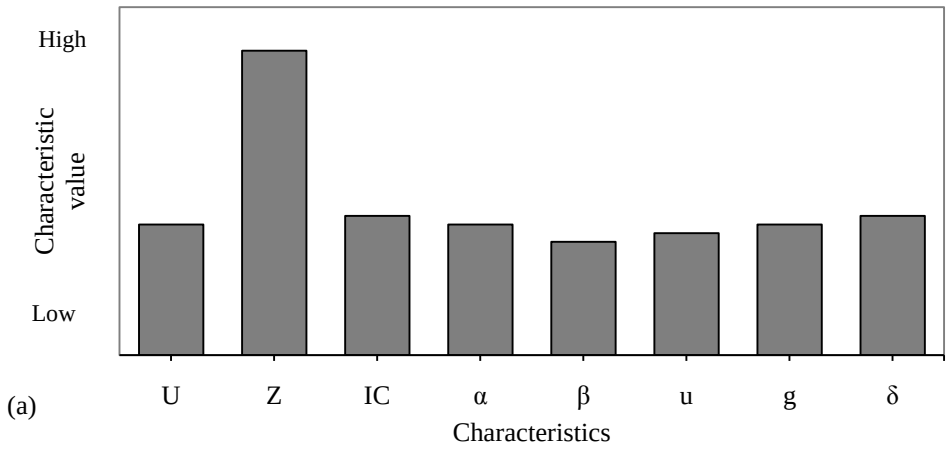
This appendix describes the detailed investigation of the solutions for 100 case studies obtained by the MMAS algorithm that led to develop ROT-b.

As described in section 10.1, the 100 case studies were randomly generated in order to represent the varying characteristics of different manufacturing systems. These characteristics including inspection errors (type I and type II), internal failure cost (rework and scrap), external failure cost (repair and replacement), inspection cost (fixed and variable), defective rates and manufacturing cost. Some of these characteristics have a greater effect on the total cost of the product than others. For example, the operation cost at processing workstations is an important characteristic for the AOIS problem. This is because the operation cost is calculated for all items processed at every processing workstation, regardless of whether or not an inspection is performed at any of the processing workstations. In addition, the defective rates generated at workstations are an important characteristic for the AOIS problem. This is because the workstations that have a high defective rate lead to an increase in the total manufacturing cost of the product. For a particular characteristic, the importance of that characteristic depends on whether its value at a workstation is high or low. For example, workstations which characteristically have a higher operating cost have more influence on the total cost of the product than the other workstations. On the other hand, workstations which characteristically have a lower unit inspection cost have more influence on the total cost of the product than the other workstations. The influence of the characteristic can be observed when an inspection station is frequently located at a workstation having that characteristic, through the inspection plans. The investigation of the inspection plans is

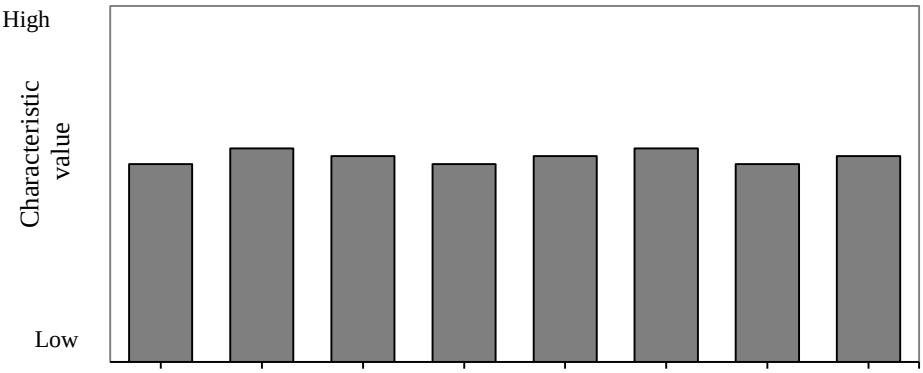
carried out by looking at where the inspection stations are located and whether the value of the characteristic at a workstation is high or low.

The aim is to find any link or relationship between the places of inspection stations and the characteristics, and to see how the MASS algorithm behaved for locating these inspection stations. In other words, the aim is to find which of these characteristics have the greatest impact on the effectiveness of the MMAS algorithms on placing inspection stations. The investigation also includes the characteristics of the processing workstations which are located after each inspection station in the inspection plans. This is because some locations of inspection stations may be influenced by such characteristics as operation cost.

Because the investigation is done manually, it is difficult to investigate all 100 cases at once. To simplify the investigation, the inspection plans of the 100 cases are divided into 10 groups in sequence, each of which consists of 10 inspection plans. The same procedure is used for their characteristics. A spreadsheet for each characteristic of the workstations for the inspection plans is prepared. The width of the spreadsheet is equal to the number of workstations. In the first inspection station of the first inspection plan, all characteristic values for workstation k (where the first inspection station is located) are checked to find out which of them has a greater influence than the others. Figure G1 shows the importance of the characteristics at workstation k in the inspection plan where the first inspection station is located. In particular, Figure G1 (a) shows that the defect rate (Z) has a greater influence than the others. Therefore, this characteristic is considered and recorded in the spreadsheet. In contrast, Figure G1 (b) shows that no characteristic has a great influence on the total cost of the product. Therefore, nothing recorded in the spreadsheet. The same procedure is used for the other inspection stations in the first inspection plan (the second inspection station, the third inspection station,..., the last inspection station). The characteristic that has the greatest influence on the MMAS for locating each inspection station in the inspection plans is recorded in the spreadsheet corresponding to the workstation. The same procedure is used for the other inspection plans. After that all these characteristics are collected and analysed.



(a)



(b)

Figure G1: Importance of the characteristics at workstation *k*

References

Abramson, D. and Abela, J. (1992). A parallel genetic algorithm for solving school timetabling problem, Appeared in 15 *Australian Computer Science Conference, Hobart,* pp.1-11.

Aghababa, M. P., Shotorbani, M. A. and Shotorbani, M. R. (2010). An adaptive particle swarm optimization applied to optimum controller design for AVR power systems, *International Journal of Computer Applications* 11(10):22–29.

Aliabadi, M., Jolai, F., Mehdizadeh, F. and Jenabi, M. (2011). A flow shop production planning problem with basic period policy and sequence dependent set up times, *Journal of Industrial and Systems Engineering,* 5(1):286-304

Angel, E. and Zissimopoulos, V. (2000). Towards a classification of combinatorial optimisation problems relatively to their difficulty for generalized local search algorithms, *Discrete Applied Mathematics*, 99, pp. 261-277.

Angus, D. (2008). *Niching Ant Colony Optimization*, PhD thesis, faculty of information & communication technologies, Swinburne University of Technology Melbourne, Australia.

Attia, A. (2005). Evidence-based medicine corner: Why should researchers report the confidence interval in modern research? *Middle East Fertility Society Journal*, 10(1):78:81.

Auger, A. and Hansen, N. (2005). A restart CMA evolution strategy with increasing population size, In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), 2, pp. 1769-1776.

Azadeh, A., Sadegh, M.S. and Amiri, A.S. (2012). A particle swarm algorithm for inspection optimisation in serial multistage processes, *Applied Mathematical Modelling,* 36(4):1455-1464.

Bai, D. and Yun, H. (1996). Optimal allocation of inspection effort in a serial multistage production system, *Computers Industrial Engineering*, 30(3): 387-396.

Ballou, D.P. and Pazer, H.L. (1982). The impact of inspector fallibility on the inspection policy in serial production systems, *Management Science*, 28(4): 387-399.

Ballou, D.P. and Pazer, H.L. (1985). Process improvement versus enhanced inspection in optimised systems, *International Journal of Production Research,* 23(6):1233-1245.

Barad, M. (1990). A break even quality level approach to location of inspection station in multistage production process, *International Journal of Production Research*, 28(1):29-45.

Baykasoğlu, A. and Dereli, T. (2009). Simple and U-type assembly line balancing by using an ant colony based algorithm, *Mathematical and Computational Applications*, 14(1):1-12.

Bell, J. E. and McMullen, P. R. (2004). Ant colony optimisation techniques for the vehicle routing problem, *Advanced Engineering Informatics,* 18, pp. 41–48.

Bentley, J. L. (1992). Fast algorithms for the geometric travelling salesman problem, ORSA *Journal on Computing*, 4, pp. 387–411.

Bianchi, L. (2006). An ant colony optimisation approach to the probabilistic travelling salesman problem, A case Study in Stochastic Combinatorial Optimisation, Universit´e Libre de Bruxelles.

Birattari, M., Paquete, L., Stützle, T. and Varrentrapp, K. (2001). Classifications of methurestics and design of experiments for the analysis of components, *Technical report* AIDA-01-05, Darmstadt University of technology, Darmstadt, Germany.

Blum, C. (2005). Beam-aco-hybridizing ant colony optimisation with beam search: Applications to open shop scheduling, *Computers and Operations Research*, 32, (6): 1565–1591.

Blum, C, Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing* Surveys, 35(3):268–308.

Boese, K. (1995). Cost versus distance in the travelling salesman problem, *Technical Report* TR-950018, UCLA CS Department.

Bui, T. N., Nguyen, T. H., Patel, C. M. and Phan, K. T. (2008). An ant-based algorithm for coloring graph, *Discrete Applied Mathematics*, 156, pp. 190–200.

Bullnheimer, B., Hartl, R. and Strauss, C. (1997). A new rank based version of the ant system, A computational Study, *Technical Report*, University of Vienna, Institute of Management Science.

Bullnheimer, B., Hartl, R.F. and Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem, *Annals of Operations Research*, 89, pp.319–328.

Campanella, J. (1990). *Principles of quality costs*, second edn, ASQ Quality Press, Milwaukee,WI.

Carson, Y. and Maria, A. (1997). Simulation optimisation: methods and application, *Proceedings of the Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, pp.118-126.

Clerc, M. and Kennedy, J. (2002). The particle swarm Explosion, stability, and convergence in a multi-dimensional complex space," *IEEE Trans. Evol. Comput.*, 6, pp. 58–73.

Chen, T. and Thornton, A. (1999). Quantitative selection of inspection plans, *Proceedings of the ASME Design Engineering Technical Conferences*, Las Vegas, Nevada.

Chengalur, I.N., Ballou, D.P. and Pazer, H.L. (1992). Dynamically determined optimal inspection strategies for serial production process, *International Journal of Production Research*, 30(1): 169-187.

Colorni, A., Dorigo, M. and Maniezzo, V. (1992). An investigation of some properties of an ant algorithm. In R.Manner and B.Manderick, editors, *Proceedings of the Parallel Problem Solving from Nature Conference* (PPSN 92), 509:520, Elsevier publishing.

Congram, K. R. (2000). *Polynomially searchable exponential neighbourhoods for sequencing problems in combinatorial optimisation*, PhD thesis, University of Southampton, UK.

Cruz, J. B., Chen, G., Li, D., and Wang, X. (2004). Particle swarm optimization for resource allocation in UAV cooperative control, *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Rhode Island.

Czogalla, J. and Fink, A. (2012). Fitness landscape analysis for the no-wait flow-shop scheduling problem, *J Heuristics*, 18, pp. 25–51.

Dianati, M., Song, I., and Treiber M. (2002). *An Introduction to Genetic Algorithms and Evolution Strategies*, Technical report. University of Waterloo, Ontario, N2L 3G1, Canada.

Deliman, C.N. and Feldman, M.R. (1996). Optimisation of process improvement and inspection location for serial manufacturing, *International Journal of Production Research*, 34(2): 395–405.

Demirkol, I., Ersoy,C., Caglayan,M. U. and Delic, H. (2001). Location area planning in cellular networks using simulated annealing, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE.

Deneubourg, J.L., Aron, S., Goss, S., and Pasteels, J.M. (1990). The self-organizing exploratory pattern of the Argentine ant, *Journal of Insect Behaviour*, 3, pp. 159–168.

Deroussi, L., Gourgand, M. and Norre, S. (2006). New effective neighbourhoods for the permutation flow shop problem, Research Report LIMOS/RR-06-09.

Di Caro, G. (2004). *Ant colony optimisation and its application to adaptive routing in telecommunication networks*, PhD thesis, faculty of sciences, Free University of Brussels, Blegium.

Di Caro, G. and Dorigo, M. (1998). Ant Net: Distributed stigmergetic control for communications networks, *Journal of Artificial Intelligence Research*, 9, pp.317–365.

Dorigo, M. and Di Caro, G. (1999). Ant algorithms for discrete optimisation, *Artificial Life*, 5(3): 137-172.

Dorigo, M. and Gambardella, L. M. (1996). A study of some properties of Ant-Q. In Voigt, H., Ebeling, W., Rechenberg, I. and Schwefel, H., editors, *Proceedings of PPSN96 International Conference on Parallel Problem Solving from Nature*, 656-665, Springer-Verlag.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the travelling salesman problem, *IEEE Transactions on Evolutionary Computation*, 1(1):53-66.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996). The ant systems: optimisation by a colony of cooperative agents, *IEEE Transaction on systems, Man, Machine and Cybernetics*, Part B, 26(1): 29-41.

Dorigo, M., Maniezzo, V. and Colorni, A. (1991a). The ant system: an autocatalytic optimising process, *Technical report*, Politecnico di Milano, Italy. No.91-016 Revised.

Dorigo, M., Maniezzo, V. and Colorni, A. (1991b). Positive feedback as a search strategy, *Technical Report* 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT.

Dorigo, M. and Stützle, T. (2002). The ant colony optimisation metaheuristic: Algorithms, applications, and advances, In Glover, F. and Kochenberger, G., editors, Handbook of Metaheuristics, vol. 57 of *International Series in Operations Research and Management Science*, pp. 251-285. Kluwer Academic Publishers, 2002.

Dorigo, M. and Stützle, T. (2004). *Ant colony optimisation*, The MIT Press, Cambridge, Massachusetts.

Duda, J. (2006). Local search and nature based metaheuristics: a case of flow-shop scheduling problem, *Proceedings of the International Multi-conference on Computer Science and Information Technology*, pp. 17–24.

Eglese, W. R. (1990). Simulated annealing a tool for operational research, *European Journal of Operational Research*, 46, pp. 271-281.

El Gamal, A.A., Hemachandra, L.A., Shperling, L. and Wei, V.K. (1987). Using simulated annealing to design good codes, *IEEE Transaction Information Theory*, IT-33, 1, pp. 116-123.

Emmons, H. and Rabinowitz, G. (2002). Inspection allocation for multistage deteriorating production systems, *IIE Transactions*, 34, pp.1034-1041.

Engin, O., Celik, A. and Kaya, I. (2008). A fuzzy approach to define sample size for attributes control chart in multistage processes: an application in engine valve manufacturing process, *Applied Soft Computing*, 8, pp.1654–1663.

Enrick, N.L. (1975). Towards optimisation of inspection allocation Part I and II, *Industrial Management*, 17(4): 7–11.

Eppen, G.D. and Hurst, G.E. (1974). Optimal location of inspection station in a multistage production process, *Management Science*, 20(8): 1194-1200.

Espinoza, F. P., Minsker, B.S. and Goldberg, D.E. (2005). Adaptive hybrid genetic algorithm for groundwater remediation design, *Journal of Water Resources Planning and Management*, 131(1): 14-25.

Ferreira, E., Goldbarg, G. and Goldbarg, M. (2012). An experimental study of variable depth search algorithms for the QAP, *Pesquisa Operationa*, 32(1): 165-195.

Fidanova, S., Marinov, P. and Atanassov, K. (2011). Sensitivity analysis of ACO starts strategies for subset problems, *Journal on Data Semantics*, 6046, PP. 256-263.

Gaertner, D. (2004). Natural algorithms for optimisation problems, *Final Year Project Report*, Imperial College, UK.

Galante, G. and Passannanti, G. (2007). Integrated approach to part scheduling and inspection policies for a job shop manufacturing system, *International Journal of Production Research*, 45(22): 5177-5198.

Gambardella, L. M. and Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the travelling salesman problem, In *International Conference on Machine Learning*, 252-60.

Gardner, L.L., Grant, M.E. and Rolston, L.J. (1995). Using simulation to assess costs of quality, in Alexopoulos, C. Kang, K. Lilegdon, R. & Goldsman, D. (Eds), *Proceedings of the Winter Simulation Conference*, Arlington, VA, IEEE Press, Piscataway, NJ, 945-951.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, 13(5):533–549.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints, *Decision Sciences*, 8, pp. 156-166.

Gluga, R, Kay, J., Lister, R. Kleitman, S. and Lever, T. (2011).Over confidence and confusion in using bloom for programming fundamentals assessment, Technical report 681, School of Information Technologies, University of Sydney.

Goksal, F., Karaoglan, I. and Altiparmak, F. (2012). A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery, *Computers & Industrial Engineering*, in Press.

Goss, S., Aron, S., Deneubourg, J. L., and Pasteels, J. M. (1989). Self-organized shortcuts in the Argentine ant, *Naturwissenschaften*, 76, pp. 579–581.

Grahl, J., Radtke, A., Minner, S. (2007). Fitness landscape analysis of dynamic multi-product lot-sizing problems with limited storage, In: Günther, H.-O., Mattfeld, D.C., Suhl, L. (eds.) Management logistischer Netzwerke, Entscheidungsunterstützung, Informations systeme und OR-Tools, pp. 257–277. Physica, Heidelberg.

Gunter, I.S. and Swanson, A.L, (1985). Inspector location in convergent production units, *International Journal of Production Research*, 23(6): 1153-1169.

Guo, Q.J. and Zheng, L. (2005). A modified simulated annealing algorithm for estimating solute transport parameters in streams from tracer experiment data, *Environmental Modelling & Software*, 20, pp. 811–815.

Hadjinicola, C.G. and Soteriou, C.A. (2003). Reducing the cost of defects in multistage production systems: a budget allocation perspective, *European Journal of Operational Research*, 145(3): 621–634.

Hamdy, A.T. (2003). *Operations research*, seventh edn, Prentice Hall.

Hansen, N. and Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions, In Xin Yao et al., editors, *Parallel Problem Solving from Nature* – PPSN VIII, LNCS 3242, pp. 282-29.1

Hansen, N., Muller, S.D. and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evol. Comput.*, 11(1):1-18.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159-195.

Hauschild, M. and Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms, Swarm and Evolutionary Computation, 1(3):111–128.

Hauskrecht, M. (2000). Value-function approximations for partially observable Markov Decision Processes, *Journal of Artificial Intelligence Research*, 13(2000): 33-94.

Hillier, F.S. H. and Lieberman, G. J. (2010). *Introduction to operations research*, ninth edn, New York, McGraw-Hill.

Hsieh, C., Chen, M. and Chen. P.(2007). Particle swarm guided evolution strategy, In Hod Lipson, editor, Genetic and Evolutionary Computation Conference (GECCO '07), pages 650:657, London, England, 7-11 July, 2007.

Hsu, S.J. (1984). A hybrid inspection system for the multistage production process, *International Journal of Production Research*, 22(1): 63-69.

Hurst, E.G. (1973). Imperfect inspection in a multistage production process, *Management Science*, 20: 378-384.

Jang, W. and Shanthikumar, J.G. (2002). Stochastic allocation of inspection capacity to competitive processes, *Naval Research Logistics*, 49(1): 78-94.

Jaszkiewicz, A. and Kominek, P. (2003). Genetic local search with distance preserving recombination operator for a vehicle routing problem, *European Journal of Operational Research* 151, pp.352–364.

Jewkes, M.E. (1995). Optimal inspection effort and scheduling for a manufacturing process with repair, *European Journal of Operational Research*, 85(2): 340-351.

Janikow, C. Z. (1993). A knowledge-intensive genetic algorithm for supervised learning, *Machine Learning*, 13, pp.189-223.

Jones, T. and Forrest, S. (1995) Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L.J. Eshelman, editor, *Proceedings of the 6th InternationalConference on Genetic Algorithms*, pages 184–192. Morgan Kaufman, San Francisco.

Jonsson, P. and Nordh, G. (2006). Generalised integer programming based on logically defined relations, *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science* (MFCS-2006), pp. 549–560.

Jupe, S.C. (2009). *Active management of distributed generation based on component thermal properties*, PhD thesis, School of Engineering and computing sciences, Durham University.

Kakade, V., Valenzuela, J. and Smith, J. (2004). An optimisation model for selective inspection in serial manufacturing systems, *International Journal of production Research*, 42(18): 3891-3909.

Kalpakjian, S. and Schmid, S. (2006). *Manufacturing engineering and technology*, fifth edn, Pearson Prentice Hall.

Kang, KS. Ebeling, A.K. and La, S. (1990). The optimal location of inspection stations using a rule-based methodology, *Computer and Industrial Engineering*, 19(1-4): 272-275.

Karaboga, D. and Okdem, S. (2004). A Simple and global optimisation algorithm for engineering problems: differential evolution algorithm, *Turk J Elec Engin*, 12(1), 53-60.

Kaya, I. and Engin, O. (2007). A New approach to define sample size at attributes control chart in multistage processes: an application in engine piston manufacturing process, *Journal Material Process Technology,* 18, pp. 38–48.

Kennedy J. and Eberhart R. C. (1995). Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks, Piscataway (1995) 1942–1948.

Kennedy, J and Mendes, R. (2002). Population structure and particle swarm performance, in *Proc. IEEE Congr. Evol. Comput.*, Honolulu, HI, pp. 1671–1676.

Kim, J. (2010). Examining the relationship between algorithm stopping criteria and performance using elitist genetic algorithm, *Proceedings of the 2010 Winter Simulation Conference*, 3220-3227.

Kirkpatrick, S., Gelatt, C.D. and Jr. (1983). Optimisation by simulated annealing, *Science,* 220(4598): 671-680.

Kogan, K. and Raz, T. (2002). Optimal allocation of inspection effort over a finite planning horizon, *IIE Transactions*, 34, pp. 515-527.

Kolahan, F. and Abachizadeh, M. (2010). Optimizing turning parameters for cylindrical parts using simulated annealing method, *International Journal of Engineering and Applied Sciences*, 6(3): 149-152.

Kumar, R. and Rockett, P. (2002). Improved sampling of the pareto front in multi-objective genetic optimisations by steady state evolution: a Pareto Converging Genetic Algorithm, In *Evolutionary* Computation, 10 (3):283 – 314.

Kumar, R., Tiwari, M. K. and Shankar, R. (2003). Scheduling of flexible manufacturing systems: An ant colony optimisation approach, *I Mech E*, pp.1443–1453.

Langner, A., Montgomery, D. and Carlyle, W. (2002). Solving a multistage partial inspection problem using genetic algorithms, *International Journal of Production Research*, 40(8): 1923-1940.

Laptik, R. and Navakauskas, D. (2009). Max-min ant system in image pre-processing, *Electronics and electrical engineering, Kaunas: Technologija*, 1(89):21–24.

Lee, J. and Chen, F.F. (1996). Inspection sequencing and part scheduling for flexible manufacturing systems, *European Journal of Operational Research*, 95(2): 344–355.

Lee, J. and Unnikrishnan, S. (1998). Planning quality inspection operations in multistage manufacturing systems with inspection errors, *International Journal of Production Research*, 36(1): 141-155.

Liao, C., Tsengb, C. and Luarn, P. (2007). A discrete version of particle swarm optimization for flow-shop scheduling problems, *Computers & Operations Research*, 34, 3099 – 3111.

Liang, Y. and Smith, A. E. (2004). An ant colony optimisation algorithm for the redundancy allocation problem, *IEEE Transactions on Reliability*, 53(3): 417-423.

Liang, Y.C., Lee, Z.S. and Chen, Y.S. (2012). A novel ant colony optimization approach for on-line scheduling and due date determination, *J Heuristics*, 18, pp: 571–591.

Lindsay, G.F. and Bishop, A.B. (1964). Allocation of screening inspection effort a dynamic programming approach, *Management Science*, 10(2): 342-352.

Mandroli, S. Shrivastava, K. and Ding, Y. (2006). A survey of inspection strategy and sensor distribution studies indiscrete-part manufacturing processes, *IIE Transactions*, 38, pp. 309-328.

Maniezzo, V. and Colorni, A. (1999). The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):769-778.

Maria, A. (1997). Introduction to modelling and simulation, Proceedings of the 1997 Winter Simulation Conference, ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, pp.7-13.

Mariano, C.E. and Morales, E. (1999). A multiple objective Ant-Q algorithm for the design of water distribution irrigation networks, *Technical Report HC-9904*, Instituto Mexicano de Tecnologı´a del Agua, Mexico.

Marmion, E. M., Jourdan, L. and Dhaenens. C. (2012). A fitness landscape analysis and metaheuristics efficiency, J Math Model Algor, pp.1-24.

McCallum, E.T. (2005). *Understanding how knowledge is exploited in ant algorithms*, PhD thesis, School of Informatics, University of Edinburgh.

Megiddo, N. (1991). Linear programming, *Encyclopaedia of Microcomputers*, 1991.

Merz, P. and Freisleben, B. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation 4, 337–352.

Merz P., Freisleben, B. (1998). Memetic algorithms and the fitness landscape of the graph bi-partitioning problem, In Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN V, pp. 765-774.

Michaelewicz, Z. and Fogel, D.B. (2000). How to solve it: Modern Heuristics, Springer-Verlag, Berlin.

Merkle, D., Middendorf, M. and Schmeck, H. (2002). Ant colony optimisation for resource-constrained project scheduling, IEEE Transactions on Evolutionary Computation, 6(4): 333-346.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953). Equation of state calculations by fast computing machines, *Journal of Chemical Physics*, 21, pp. 1087–1092.

Mills, P., Tsang, E. and Ford, F. (2003). Applying an extended guided local search to the quadratic assignment problem, *Annals of Operations Research*, 118, pp.121:135.

Moeini, R. and Afshar, M.H. (2009). Application of an ant colony optimisation algorithm for optimal operation of algorithm for optimal operation of reservoirs: a comparative study of three proposed formulations, *Transaction A: Civil Engineering*, 16(4):273-285.

Montgomery, D.C. (1997). *Introduction to statistical quality control*, third edn, New York, John Wiley & Sons.

Muller, C. and Sbalzarini, I. (2011). Global characterization of the CEC 2005 fitness landscapes using fitness-distance analysis, EvoApplications'11 Proceedings of the 2011 international conference on Applications of evolutionary computation-Volume Part I, pp. 294-303.

Nahas, N. and Nourelfath, M. (2005). Ant system for reliability optimisation of a series system with multiple-choice and budget constraints, *Reliability Engineering and System Safety*, 87, pp. 1–12.

Narahari, Y., and Khan, L.M. (1996). Modelling re-entrant manufacturing systems with inspection stations, *Journal of Manufacturing Systems*, 15(6): 367–378.

Ning, X., Lam, KC. and Lam, MC. (2010). Dynamic construction site layout planning using max-min ant system, *Automation in Construction*, 19, pp. 55–65.

Ochoa, G., Rodriguez, J., Petrovic, S. and Burke, E. (2009). Dispatching rules for production scheduling: a hyper-heuristic landscape analysis, IEEE, pp. 1873-1880.

Omidvar, M. and Li, X. (2011). A comparative study of CMA-ES on large scale global optimisation, *AI 2010: Advances in Artificial Intelligence*, pp. 303–312.

Önüt, S., Tuzkaya, R and Dogac, B. (2008). A particle swarm optimization algorithm for the multiple-level warehouse layout design problem, *Computer Industrial Engineering*, 54 pp. 783–799.

Osman, I.H. and Laporte, G. (1996). Metaheuristics: A bibliography, *Annals of Operations Research*, 63, pp. 513–628.

Ostwald, P. F. and McLaren, T. S. (2004). *Cost Analysis and Estimating for Engineering and Management*, Prentice Hall.

Ozcan, E. and Mohan, C. (1999). Particle swarm optimization: Surfing the waves, in *Proc. IEEE Congress Evol. Comput.*, 3, pp.1939–1944.

Park, H.Y., Park, H.E. and Ntuen, A.C. (1988). A study for optimal inspection policies in a flexible manufacturing cell, *Computer and Industrial Engineering*, 15(1-4): 307-314.

Pellegrini, P. and Moretti, E. (2009). A computational analysis on a hybrid approach: quick-and-dirty ant colony optimization, *Applied Mathematical Sciences*, 23(3): 1127–1140.

Penn, M. and Raviv, T. (2007). Optimising the quality control station configuration, *Naval Research Logistics*, 54, pp. 301-314.

Peters, M.H. and Williams, W.W. (1984). Location of quality inspection stations: an experimental assessment of five normative heuristics, *Decision Sciences*, 15(3): 389-408.

Prakash, A. Tiwari, M. K. and Shankar, R. (2008). Optimal job sequence determination and operation machine allocation in flexible manufacturing systems: an approach using adaptive hierarchical ant colony algorithm, *Journal of Intelligent Manufacturing*, (19), pp.161–173.

Pruzan, P.M. and Jackson, J.T.R. (1967). A dynamic programming application in production line inspection, *Techno-metrics*, 9(1): 73-81.

Raghavachari, M. and Tayi, G. (1991). Inspection configuration and reprocessing decisions in serial production systems, *International Journal of Production Research*, 29(5): 897-911.

Rajab, S.R. (2012). *Some applications of continuous variable neighbourhood search metaheuristic* (mathematical modelling), PhD thesis, Brunel University, UK.

Ram, J., Sreenivas, H. and Subramaniam, G. (1996). Parallel simulated annealing algorithms. *Journal of Parallel and Distributed Computing*, 37, pp. 207-212.

Rau, H. and Chu, YH. (2005). Inspection allocation planning with two types of workstations WVD and WAD, *International Journal of Advanced Manufacturing Technology*, 25(9): 947-953.

Rau, H., Chu, YH. and Cho, KH. (2005). Layer modelling for the inspection allocation problem in re-entrant production systems, *International Journal of Production Research*, 43(17): 3633-3655.

Raz, T. (1986). A survey of models for allocating inspection effort in multistage production system, *Journal of Quality Technology*, 18(4): 239-247.

Raz, T. and Bricker, D. (1987). Sequencing of imperfect inspection operation subject to constraints on the quality of accepted and rejected units, *International Journal of Production Research*, 25(6): 809-821.

Raz, T. and Kaspi, M. (1991). Location and sequencing of imperfect inspection in serial multistage production systems, *International Journal of Production Research*, 29(8): 1645-1659.

Rechenberg, I. (1964). Kybernetische Losungsansteuerungeiner experimentellen Forschungsaufgabe, presented at the Annual Conference of the WGLR at Berlin inSeptember.

Reeves, C.R., (1993). Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications.

Reeves, C. R. (1998). Landscapes, operators and heuristic search, *Annals of Operations Research*, 86(0): 473-490.

Reyes, M. and Coello, C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art, *International Journal of Computational Intelligence Research*, 3(2):287-308.

Rezazadeh, H., Ghazanfari, M. and Mehrabad, M. (2009). An extended discrete particle swarm optimization algorithm for the dynamic facility layout problem, *Journal of Zhejiang University SCIENCE A*, 10(4): 520-529.

Ridge, E. (2007). *Design of experiments for the tuning of optimisation algorithms*, PhD thesis, department of computer science, The University of York.

Ridley, A. (2008). Allocation of inspection effort for Rolls Royce Trent 700 turbine blade manufacture, *MSc. Dissertation thesis*, Durham University, UK.

Ritchie, G. (2003). S*tatic multi-processor scheduling with ant colony optimisation & local search*, Master dissertation, School of Informatics, University of Edinburgh.

Rodchua, S. (2006). Factors, measures, and problems of quality costs program implementation in the manufacturing environment, *Journal of Industrial Technology*, 22(4): 1-6.

Rossi, A. and Dini, G. (2007). Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method, *Robotics and Computer-Integrated Manufacturing*, 23(5): 503-516.

Sadegheih, A. (2007). Sequence optimisation and design of allocation using GA and SA, *Applied Mathematics and Computation*, 186(2): 1723-1730.

Saxena, S., Chang, C.M., Chow, H.B. and Lee, J. (1990). Evaluation of heuristics for inspection station allocation in serial production systems, *Proceedings of the 1990 Winter Simulation Conference*, pp. 919-922.

Schwefel, H. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Str¨omungstechnik*, PhD thesis, Technical University of Berlin, Hermann F¨ottinger–Institute for Hydrodynamics.

Selvi, V. and Umarani, R. (2010). Comparative analysis of ant colony and particle swarm optimization techniques, *International Journal of Computer Applications*, 5(4), pp: 1-6

Siemiatkowski, M. and Przybylski, W. (2006). Simulation studies of process flow with in-line part inspection in machining cells, *Journal of Materials Processing Technology*, 171(1): 27–34.

Silva, C. A., Runkler, T. A., Sousa, J. M. and Palm, R. (2002). Ant colonies as logistic process optimizers, In: Dorigo M, Di Caro G, Samples M, editors. Ant algorithms *Proceedings of ANTS 2002*, Third international workshop. Lecture Notes in Computer Science, 2463. Berlin: Springer, pp. 76–87.

Silver, E. (2004). An overview of heuristic solution methods, *Journal of the Operational Research Society* 55, pp. 936–956.

Shan, Y. McKay, R. I. Essam, D. and Abbass, H. A. (2006). A survey of probabilistic model building genetic programming. In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Springer, 2006.

Shetwan, A., Vitanov, V. and Tjahjono, B. (2011). Allocation of quality control stations in multistage manufacturing systems, *Computers and Industrial Engineering*, 60(4), 473-484.

Shi, Y. and Eberhart, R. (1998). Parameter selection in particle swarm optimization, in *Proc. 7th Int. Conf. Evol. Program.* (*EP*), pp. 591–600.

Shi, Y. and Eberhart, R. (1999). Empirical study of particle swarm optimization, in *Proc. IEEE Congr. Evol. Comput.*, 3, pp.1945–1950

Shiau, YR. (2002). Inspection resource assignment in a multistage manufacturing system with an inspection error model, *International Journal of Production Research*, 40(8): 1787-1806.

Shiau, YR. (2003a). Inspection allocation planning for a multiple quality characteristics advanced manufacturing technology, *International Journal of Advanced Manufacturing Technology*, 21(7): 494-500.

Shiau, YR. (2003b). Quick decision-making support for inspection allocation planning with rapidly changing customer requirements, *International Journal of Advanced Manufacturing Technology*, 22(9-10): 633-640.

Shiau, YR., Lin, MH. and Chung, WC. (2007). Concurrent process inspection planning for a customized manufacturing system based on genetic algorithm, *International Journal Advanced manufacturing Technology*, 33(7-8): 746-755.

Shin, W.S., Hart, S.M. and Lee, H.F. (1995). Strategic allocation of inspection stations for a flow assembly line: a hybrid procedure, *IIE Transactions*, 27, pp. 707–715.

Shuang, B., Chenand, J. and Li, Z. (2011). Study on hybrid PS-ACO algorithm, *Applied Intelligence*, 34(1): 64-73.

Sim, E., Koh, S.P., Chen, C.P., Tiong, S.K. and Fong, A. (2011). Optimisation of dispensing parameter fine tuning program using genetic algorithm, *International Conference on Computer Engineering and Applications IPCSIT vol.,2 IACSIT Press, Singapore, 163-168.*

Smit, S. and Eiben, A. (2011). Parameter tuning of evolutionary algorithms: generalist vs. specialist, *Applications of Evolutionary Computation*, pp. 542–551.

Socha, K., Knowles, J. D. and Sampels, M. (2002). A max-min ant system for the university course timetabling problem, *Proceedings of the 3rd International Workshop on Ant Algorithms, ANTS 2002*, Lecture Notes in Computer Science, 2463, Springer, pp.1–13.

Socha, K., Sampels, M. and Manfrin, M. (2003). Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. *Proceedings of EvoCOP* 2003–3rd European Workshop on Evolutionary Computation in Combinatorial Optimisation, volume 2611 of *LNCS*, pp. 334-345, Springer-Verlag.

Solimanpur, M., Vrat, P. and Shankar, R. (2004). Ant colony optimisation algorithm to the inter-cell layout problem in cellular manufacturing, *European Journal of Operational Research*, 157, pp. 592–606.

Solimanpur, M., Vrat, P. and Shankar, R. (2005). An ant algorithm for the single row layout problem in Flexible manufacturing systems, *Computers & Operations Research*, 32, pp. 583–598.

Spiliopoulos, K. and Sofianopoulou, S. (2008). An efficient ant colony optimisation system for the manufacturing cells formation problem, *International Journal Advanced manufacturing Technology*, 36, pp.589–597.

Stadler, F.P. (1996). Landscapes and their correlation functions, *Journal of Mathematical Chemistry*, 20, pp.1–45.

Storn, R. and Price, K. (1995). Differential Evolution- a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, *Technical Report* TR-95-012, International Computer Science Institute, Berkley.

Stützle, T. (1997). Max-min ant system for quadratic assignment problems, *Technical Report*, AIDA.97.4, FG Intellektik, TU Darmstadt, Germany.

Stützle, T. (1998a). An ant approach to the flow shop problem. In *Proceedings of EUFIT'98*, pp. 1560.1564, Aachen.

Stützle, G. T. (1998b). Local search algorithms for combinatorial problems: analysis, improvements, and new applications, 220 of DISKI. Sankt Augestin, Germany, Infix.

Stützle, T. (2003). Iterated local search variable neighbourhood search, *MN summer school*, Tenerife.

Stützle, T. and Dorigo, M. (1999). ACO algorithms for the travelling salesman problem. In Miettinen, K., Makela, M., Neittaanmaki, P., and Periaux, J., editors, *Evolutionary Algorithms in Engineering and Computer Science*, pp. 163.183, Wiley.

Stützle, T. and Hoos, H. (1997). The max-min ant system and local search for the travelling salesman problem, *Proceedings of ICEC'97*, IEEE fourth edn, International Conference on Evolutionary Computation, IEEE Press, pp. 308–313.

Stützle, T. and Hoos, H. (1998). Improvements on the ant system: Introducing the max-min ant system, In Smith, G., Steele, N., and Albrecht, R., editors, *Proceedings of Artificial Neural Nets and Genetic Algorithms 1997*, pp. 245.249.Springer Verlag Wien.

Stützle, T. and Hoos, H. (2000). Max-min ant system, *Future Generation Computer Systems*, 16, pp. 889–914.

Stützle, T., Lopez, M. and Paola, P. (2010). Parameter adaptation in ant colony optimisation, *Technical Report*, Series, No.TR/IRIDIA/2010-002.

Suman, B. and Kumar, P. (2006). A survey of simulated annealing as a tool for single and multi-objective optimization, *Journal of the Operational Research Society*, 57, 1143–1160.

Taillard, E. D. and Gambardella, L. (1997). An ant approach for structured quadratic assignment problems, *Technical report*, IDSIA-22-97, IDSIA.

Taneja, M. and Viswanadham, N. (1994). Inspection allocation in manufacturing systems a genetic algorithm approach, *Proceedings of the IEEE California International Conference on Robotics and Automation*, (San Diego, California, 8-13 May),pp. 3537-3542.

Tayi, G.K. and Ballou, D.P. (1988). An integrated production inventory model with reprocessing and inspection, *International Journal of production Research*, 26(8): 1299-1315.

Teytaud, F. (2011). *Introduction of Statistics in Optimization*, PhD thesis, University Paris-Sud 11.

Thepsonthi, T. and Özel, T. (2012). Multi-objective process optimization for micro-end milling of Ti-6Al-4V titanium alloy, *International Journal of Advanced Manufacturing Technology*, 63, pp. 903–914.

Thiesen, A. (1998). Design and evaluation of tabu search algorithms for multiprocessor scheduling, *Journal of Heuristics*, 4, pp.141–160.

Ursem, R. (2003). *Models for evolutionary algorithms and their applications in system identification and control optimization*, PhD thesis, department of computer science, University of Aarhus, Denmark.

Valenzuela, F.J., Smith, S.J. and Evans, S.J. (2004). Allocating solder-paste printing inspection in high volume electronics manufacturing, *IIE Transactions*, 36, pp. 1171–181.

Vanitha, M. and Thanushkodi, K. (2011). Solution to economic dispatch problem by differential evolution algorithm considering linear equality and inequality constrains, *International Journal of Research and Reviews in Electrical and Computer Engineering*, 1(1):21-26.

Van Volsem, S., Dullaert, W. and Van landeghem, H. (2007). An evolutionary algorithm and discrete event simulation for optimising inspection strategies for multistage processes, *European Journal of Operational Research*, 179(3): 621-633.

Van Volsem, S. and Neirynck, S. (2009). Adapting an evolutionary algorithm with embedded simulation and pseudo-random number generation for the cell broadband engine, *European Simulation and Modelling Conference*, October,26-28, Leiceter,UK.

Visa, S., Inoue, A. and Ralescu, A. (2011). Plenary lecture, Proceedings of the twenty second Midwest artificial intelligence and cognitive science conference, April 16–17, 2011 University of Cincinnati, Ohio.

Viswanadham, N., Sharma, M.S. and Taneja, M. (1996). Inspection allocation in manufacturing systems using stochastic search techniques, *IEEE Transactions on systems*, Man and cybernetics, Part A: Systems and human, 26(2): 222–230.

Wang, J. (2007). Petri Nets for dynamic event-driven system modelling, handbook for dynamic system modelling, Ed: Paul fishwick, CRC Press.

Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference, *Biological Cybernetics*, 63(5):325–336.

Weise, T. (2009). *Global optimisation algorithms theory and application*, second edn, Newest Version: http:/www.it-weise.de/.

White, L.S. (1965). The analysis of a simple class of multistage inspection plans, *Management Science*, 12, (9): 685-693.

White, L.S. (1969). Shortest route models for the allocation of inspection effort on production line, *Management Science*, 15(5): 249-259.

Wierstra, D., Schaul, T., Peters, J. and Schmidhuber, J. (2008). Natural evolution strategies, accepted at *IEEE World Congress on Computational Intelligence (WCCI 2008)*.

Wild, R. (1989). *Production and operations management*, fourth edn, 1989.

Wong, Y. K. and See, C. P. (2009). A new minimum pheromone threshold strategy (MPTS) for max–min ant system, *Applied Soft Computing*, 9, pp. 882–888.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution, In: D. Jones (ed.) International Proceedings of the Sixth International Congress on Genetics, 1, pp. 356–366.

Wu, K., Ting, J. and Gonzalez, L. (2011). An ant colony optimisation algorithm for quadratic assignment problem, Proceedings of 12th Asia Pacific Industria1Engineering: and Mana2ement Systems Conference. Beijing. China. October 14-16, pp. 138-144.

Yamada, T. (2003). *Studies on Metaheuristics for Job-hop and Flow-shop Scheduling Problems*, PhD thesis, Kyoto University, Kyoto, Japan.

Yin, Y., Yu, S., Wang, P. and Wang, T. (2006). A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems, *Computer Standards & Interfaces*, 28, 441–450.

Yuan, B. and Gallagher. M. (2005). A hybrid approach to parameter tuning in genetic algorithms. In Proceedings of the IEEE Congress in Evolutionary Computation (CEC'05), volume 2, pages1096–1103. IEEE Press, Piscataway, NJ, 2005.

Yum, J.B. and McDowell, D.E. (1987). Optimal inspection policies in serial production system including scrap rework and repair: An MILP approach, *International Journal of Production Research*, 25(10): 1451-1464.

Zhang, Q., Sun, J., Tsang, E. and Ford, J. (2003a). Hybrid estimation of distribution algorithm for global optimisation, *Engineering Computations*, 21(1), pp 91-107.

Zhang, Q., Sun, J., Tsang E. and Ford, J. (2003b). Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem, In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 42-48.

Zhang, Q., Sun, J. and Tsang, E. (2004). Evolutionary algorithm with guided mutation for the maximum clique problem, IEEE *Transactions on Evolutionary Computation*, 9 (2), 192-200.

Zhi-He, W. (2008). Settings of algorithm parameters in ant colony algorithm, *International Conference on Computer Science and Information Technology*, pp. 724-728.

Publications

1. Shetwan, A., Vitanov, V. and Tjahjono, B. (2011). Allocation of quality control stations in multistage manufacturing systems, *Computers and Industrial Engineering*, 60(4), 473-484.
2. Shetwan, A., and Vitanov, V. (2009). Optimum allocation of inspection effort in multistage manufacturing process. *European Simulation and Modelling Conference*,October,26-28, Leiceter,United Kingdom.
3. Shetwan, A., and Vitanov,V. (2010). Allocation of inspection effort in multistage manufacturing process using ant colony optimisation. *21st International Computer-Aided Production Engineering Conference*, April, 13-14, University of Edinburgh, UK.
4. Shetwan, A., and Vitanov, V. (2010). Allocation of inspection effort in multistage manufacturing process using ant colony with local search. *8th International Conference in Manufacturing Research*, 14-16, September, Durham University, UK.

5. Shetwan, A., and Vitanov, V. (2011). Location of inspection stations in multistage manufacturing processes, Fubutec2011, *Future Business Technology*, 18-20, April, British Institute of Technology and Ecommerce, London, UK.